Learning Decision-Focused Uncertainty Representations: Theory and Applications in Sustainability

Thesis by
Christopher Tzong-Ran Yeh

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2026
Defended May 22, 2026

Christopher Tzong-Ran Yeh
ORCID: 0000-0002-7624-6168

ACKNOWLEDGEMENTS

I am indebted to my advisors, Yisong Yue and Adam Wierman, for their unwavering support, guidance, and mentorship throughout my PhD journey. Their confidence in me consistently pushed me to do my best work. I especially thank Yisong for encouraging me to always think about the bigger picture, to look for the important "step functions" in a research trajectory, and to keep sight of the broader vision. He helped me form and articulate that vision, connected me with collaborators and resources, and gave me the freedom to pursue the questions that most interested me. Adam has likewise been a steady source of support, especially through his commitment to work that matters for energy and sustainability, his help in getting my first PhD project off the ground, and his insistence on technical rigor, which shaped the way I think about research.

I am also grateful to my committee members, Steven and Priya, for their thoughtful feedback at key moments in my research career. Both have been role models to me, and both deepened my interest in energy and sustainability applications.

Several collaborators deserve special thanks. Yuanyuan Shi and Jing Yu provided tremendous support on my first PhD project, which focused on voltage control. When I was new to both power systems and online learning, Yuanyuan generously shared her time and expertise with me and helped me work through two full iterations of the problem formulation before we finally arrived at the right one. Jing also offered invaluable technical and moral support, and helped me get started learning basic control theory.

My PhD research has also benefited greatly from my collaboration with Nicolas (Nico) Christianson. More times than I can count, Nico provided the insight that turned a mediocre project into a strong one. His expertise in power systems, online algorithms, and optimization has been pivotal to many of my works, and working with him late into the night to meet deadlines has been one of the highlights of my PhD experience.

I have learned so much from my other collaborators as well: Yiheng Lin, Zaiwei Chen, Tongxin Li, Han Xu, Rajeev Datta, Victor Li, Julio Arroyo, Apoorva Thanvantri, Lingkai Kong, Kai Wang, Zihao Zhao, and many more.

I am thankful for the many friends and colleagues at Caltech who have enriched my time here through research discussions and friendship, including Eitan Levin, Pau

Battle, Lauren Conger, John Lathrop, Margaret Trautner, Lucien Werner, Laixi Shi, Kishan Panaganti, Yiheng Xie, Chengrui Qu, Jennifer Sun, Francesca-Zhoufan Li, Jason Yang, Sabera Talukder, Geeling Chau, Xuefei (Julie) Wang, Raul Astudillo, Jeremy Bernstein, Uriah Israel, and the rest of the Yue Crew and RSRG/FALCON groups.

Beyond research, I would like to thank the CMS administrative staff, especially Sumaia Abedin, Jolene Brink, Alma Rangel-Fuentes, and Christine Ortega, for their steady support with both large and small matters. I also thank Neil Fromer for connecting me with the Resnick Sustainability Institute and the broader sustainability research community at Caltech.

Finally, I owe my deepest gratitude to my family. My mother, Chia, made countless sacrifices throughout my childhood to support my education, and my father, Henry, first introduced me to the Caltech campus during a sabbatical visit in 2011. My brother, Ben, has been my closest confidant and the backstop of my life; he has shown me what grit and perseverance look like, and he has been a constant source of encouragement and support.

This thesis is dedicated to every teacher, mentor, and educator who has supported me on my educational journey.

ABSTRACT

Managing risks arising from decision-making under uncertainty is a paramount challenge in many sustainability and energy applications. The threat of climate change paired with the increasing complexity of energy systems has made it more important than ever to develop artificial intelligence (AI) and machine learning (ML) methods that can address these challenges. However, standard AI and ML methods suffer from a trade-off between accuracy and reliability. While modern AI models are increasingly capable and accurate, they generally lack rigorous uncertainty quantification for reliability guarantees. On the other hand, traditional methods for uncertainty quantification often rely on strong assumptions and are not designed to optimize decision quality.

This thesis develops new methods for overcoming this trade-off by learning uncertainty representations that are directly useful for decision-making. The first part of the thesis studies batch decision problems, where a model is trained on historical data and then used to make one-shot decisions. A central theme is that good predictions alone are not enough: the uncertainty must be shaped to match the downstream task. Instead, we develop a general framework for decision-focused learning of uncertainty representations, which can be tailored to a wide range of optimization problems and risk guarantees. By leveraging differentiable optimization, we develop end-to-end methods for learning calibrated confidence scores for achieving tail-risk guarantees, uncertainty sets for robust optimization, and generative models for Wasserstein distributionally robust optimization. We also extend decision-focused learning to diffusion models so that full predictive distributions can be used for stochastic optimization. Across energy storage, generator scheduling, and related applications, these methods improve robustness and decision quality while retaining theoretical guarantees or tractable optimization.

The second part of the thesis turns to sequential decision-making. For voltage regulation in distribution grids with uncertain topology, the thesis develops an online control algorithm with a finite-mistake guarantee, showing that safe control can be achieved even while the system is still being learned. The thesis also introduces SustainGym, a benchmark suite of reinforcement learning environments for sustainability applications, including electric vehicle charging, electricity markets, datacenters, cogeneration, and building control. SustainGym exposes realistic distribution shifts, physical constraints, and multi-agent interactions that are often absent from standard

benchmarks, and it reveals how brittle off-the-shelf reinforcement learning (RL) methods can be in these settings. We develop a new family of distributionally robust RL algorithms that can learn policies with strong out-of-distribution generalization guarantees, and we show that these methods can achieve superior performance in SustainGym and game-playing environments.

Taken together, the thesis argues for a general principle: uncertainty should be learned, calibrated, and evaluated in terms of the decisions it enables. By connecting uncertainty quantification, optimization, online learning, and reinforcement learning, this work provides both theory and practical tools for building reliable decision systems in sustainability-critical applications.

PUBLISHED CONTENT AND CONTRIBUTIONS

[209] Chengrui Qu, Christopher Yeh, Kishan Panaganti, Eric Mazumdar, and Adam Wierman. 2026. Distributionally robust cooperative multi-agent reinforcement learning with value factorization. In *The Fourteenth International Conference on Learning Representations*. (Apr. 2026). https://openreview.net/forum?id=2T3LOpqIOO.

C.Y. participated in the conception of the project, advised in the development of the theoretical results and algorithms, advised the conception and implementation of experiments, and particpated in the writing of the manuscript.

[323] Zihao Zhao, Christopher Yeh, Lingkai Kong Kong, and Kai Wang. 2026. Diffusion-DFL: decision-focused diffusion models for stochastic optimization. In *The Fourteenth International Conference on Learning Representations*. (Apr. 2026). https://openreview.net/forum?id=uhv3f80jmG.

C.Y. helped develop the theoretical results and algorithms, advised the conception and implementation of experiments, and particpated in the writing of the manuscript.

[295] Han Xu and Christopher Yeh. 2025. Robust energy storage operation via generative Wasserstein distributionally robust optimization. In *NeurIPS 2025 Workshop on Tackling Climate Change with Machine Learning*. San Diego, CA, USA, (Dec. 2025). https://www.climatechange.ai/papers/neurips2025/79.

C.Y. helped with the conception of the project, participated in the development of the algorithms, and participated in the writing of the manuscript.

[306] Christopher Yeh, Nicolas Christianson, Adam Wierman, and Yisong Yue. 2025. Conformal Risk Training: End-to-End Optimization of Conformal Risk Control. In *Advances in Neural Information Processing Systems*. Vol. 38. San Diego, CA, USA, (Dec. 2025), 70056–70092. https://proceedings.neurips.cc/paper_files/paper/2025/hash/6559542f75b4452ebaaf82094c7defb7-Abstract-Conference.html.

C.Y. led the conception of the project and the development of the theoretical results and algorithms, conducted the simulations, and led the writing of the manuscript.

[311] Christopher Yeh*, Nicolas Christianson*, Alan Wu, Adam Wierman, and Yisong Yue. 2025. End-to-end conformal calibration for optimization under uncertainty. *Transactions on Machine Learning Research*, (Dec. 2025). https://openreview.net/forum?id=yM8qkT0f9H.

C.Y. co-led the conception of the project and development of the theoretical results and algorithms, was responsible for the conception and conduction of most simulations, and co-led the writing of the manuscript.

[307] Christopher Yeh, Jing Yu, Yuanyuan Shi, and Adam Wierman. 2024. Online learning for robust voltage control under uncertain grid topology. *IEEE Transactions on Smart Grid*, 15, 5, (Sept. 2024), 4754–4764. doi:10.1109/TSG.2024.3383804.

C.Y. led the conception of the project and the development of the theoretical results and algorithms, conducted the simulations, and led the writing of the manuscript.

[310] Christopher Yeh et al. 2023. SustainGym: reinforcement learning environments for sustainable energy systems. In *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., New Orleans, LA, USA, (Dec. 2023), 59464–59476. https://proceedings.neurips.cc/paper_files/paper/2023/hash/ba74855789913e5ed36f87288af79e5b-Abstract-Datasets_and_Benchmarks.html.

C.Y. led the conception of the project, participated in the development of the simulations, and led the writing of the manuscript.

[308] Christopher Yeh, Jing Yu, Yuanyuan Shi, and Adam Wierman. 2022. Robust online voltage control with an unknown grid topology. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*. Association for Computing Machinery, (June 2022), 240–250. ISBN: 9781450393973. doi:10.1145/3538637.3538853.

C.Y. led the conception of the project and the development of the theoretical results and algorithms, conducted the simulations, and led the writing of the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Number *Page*

LIST OF TABLES

Chapter 1

INTRODUCTION

Because we almost never have perfect information about the environment in which we, or systems that we design, act and behave, decision-making under uncertainty is ubiquitous. In real-time operations of the energy grid, for example, grid operators face significant uncertainty in both energy demand and supply. Demand can vary with industrial activity, weather, and other exogenous factors, while supply can vary with weather-driven uncertainties in wind and solar generation. When optimizing grid-scale battery storage operations, uncertainty in future energy prices requires risk management strategies to maximize expected utility without incurring excessive risk [6]. For scheduling compute jobs, data centers themselves must manage uncertainty in future workloads while seeking to maintain certain levels of performance and to minimize their carbon footprint [214, 118]. While this thesis primarily focuses on uncertainty in sustainability and energy applications, the challenges of decision-making under uncertainty are likewise apparent in other domains; where appropriate, we occasionally refer to medical, financial, and game-playing applications.

Notably, the types of uncertainties our systems confront are often growing in complexity and scale. Recent evidence suggests that anthropogenic climate change itself is accelerating [88], so addressing sustainability challenges is evermore urgent yet challenging. Furthermore, energy systems are becoming more complex, with more distributed energy resources, renewable generation, and controllable loads. Part of the challenge is that, with more data available, we design more complex systems, and we expect our systems to handle correlated uncertainties.

Over the past decade, artificial intelligence (AI) and machine learning (ML) methods have achieved significant breakthroughs across a wide range of domains, including game-playing [178, 245, 137], visual understanding [134], and natural language understanding [45]. These methods have also been increasingly applied to decision-making problems in sustainability applications such as energy systems [195, 206, 312], datacenter scheduling [214, 118], and building control [316, 273, 318]. These successes suggest that AI/ML methods hold promise as powerful tools for addressing uncertainty.

Nonetheless, existing methods for uncertainty quantification still suffer from two

closely related limitations. First, they often do not come with guarantees that are broadly applicable to modern AI tools, especially when the downstream objective depends on tail behavior, robustness, or constraint satisfaction rather than on average prediction error alone. Second, systems today largely separate uncertainty quantification from decision-making, treating the two as separate steps even when the relevant uncertainty representation should depend on the task itself. This separation can be particularly problematic in high-stakes settings, where a well-calibrated predictor may still induce poor decisions if its uncertainty is shaped incorrectly, or where a deterministic predictor fails to capture the variability that matters for the task [55, 123, 256, 203, 284, 246, 237, 119, 112].

This thesis seeks to address both of these shortcomings. First, how can we establish rigorous uncertainty quantification and risk control guarantees, suitable for modern AI tools? We seek to make minimal assumptions about the data and the AI model to achieve widely applicable guarantees. Second, and more importantly, how can we *co-design* how we do uncertainty quantification *for* decision-making? The trade-off for making minimal assumptions is that the guarantees can naively be quite weak. So instead of making assumptions on the data or AI model, can we leverage information about the structure of the decision-making task? That is, we propose *learning* decision-focused uncertainty representations optimized for the target task.

1.1 Challenges and Prior Work

Uncertainty quantification. Uncertainty quantification is widely studied across the field of ML, and a broad range of metrics has been developed to characterize the uncertainty of model predictions. For example, probabilistic machine learning places considerable emphasis on *calibration*. Whereas it has long been known that loss functions based on proper scoring rules should yield calibrated models [96], this theoretical guarantee is only valid in the limit of infinite training data and globally optimal optimization algorithms. Moreover, calibration alone is not sufficient, because a model can be perfectly calibrated while still having little or no predictive value if it merely reproduces marginal statistics.

Related work has also focused on quantifying uncertainty through prediction intervals, or more generally, prediction sets, which are intended to contain the ground truth labels with high probability. This is known as achieving marginal *coverage*. Conformal prediction and its distribution-free finite-sample guarantees [278, 233, 9] have made important progress in this direction, but many methods are still evaluated primarily

by nominal coverage or set size rather than by their effect on the downstream decision. Two models can have the same coverage rate while inducing very different actions, and a slightly larger prediction set may be harmless in one application and harmful in another. Likewise, a method that improves coverage can still degrade the final decision if the uncertainty set is poorly aligned with the optimizer that uses it. The thesis therefore treats uncertainty representations as decision objects, not as auxiliary diagnostics.

Decision-focused learning. A separate line of work studies predict-then-optimize and decision-focused learning (DFL) [73, 78, 175, 226]. These methods recognize that prediction error is not always aligned with decision quality, and they differentiate through the downstream optimization problem so that training can target the actual task loss. This is a decisive improvement over purely predictive training when the eventual action depends on the structure of the decision problem.

However, much of the DFL literature still relies on deterministic point estimates. In applications where the downstream cost is sensitive to tails, ambiguity, or rare events, point estimates are often too impoverished. A point prediction can be accurate on average while still producing a risky or conservative decision. Conversely, a probabilistic prediction may be well calibrated but not shaped in a way that supports the downstream objective. The thesis develops methods that retain the decision-aware spirit of DFL while incorporating calibrated uncertainty, risk control, or learned generative models.

Online learning and reinforcement learning. Uncertainty also appears in sequential decision-making, where the controller acts repeatedly and the environment may change over time. Online learning and reinforcement learning (RL) naturally address this type of setting, but their guarantees are often weaker than the finite-sample or robustness guarantees available in the batch decision literature. In practice, RL benchmarks are also frequently too idealized to stress-test algorithms in realistic sustainability settings. This thesis therefore studies sequential control through two complementary lenses: an online voltage control problem with explicit stability guarantees, and a benchmark suite for sustainability-focused RL that exposes realistic distribution shifts, physical constraints, and multi-agent interaction.

Bayesian decision-making. The Bayesian framework is another classical response to uncertainty. It begins with a prior over the data-generating process, updates

Table 1.1: Thesis overview

	Chapter	decision-making task	uncertainty representation	risk / robustness guarantee
batch	§2	plug-in optimization	scalar confidence score	tail risk control
	§3	robust optimization	uncertainty (prediction) set	marginal coverage
	§4	Wasserstein DRO	ambiguity (credal) set	(none)
	§5	stochastic optimization	predictive distribution, scenario samples	(none)
online	§6	voltage control	uncertainty set	finite mistake
	§8	reward maximization	ambiguity set	Q-value lower bound

to a posterior after observing data, and then computes a Bayes-optimal decision under that posterior. This is conceptually elegant and often statistically powerful. In large-scale sustainability systems, however, the choice of prior can be difficult to justify, and posterior computation or downstream Bayes-optimal optimization can quickly become expensive. For the purposes of this thesis, a frequentist perspective is more appropriate: it allows distribution-free guarantees, makes minimal assumptions, and keeps the method focused on the operational decision rather than the subjective prior.

1.2 Thesis Contributions

Table 1.1 summarizes the structure of the thesis. The thesis is split into two parts. Part 1 studies the batch decision setting, where a model is trained on a dataset and then used to make one-shot decisions. Part 2 studies the online setting with sequential decisions.

Part 1: batch decision-making

In Part I, we focus on decision problems that belong to the predict-then-optimize paradigm [73, 291, 78, 175, 226] which encompasses a wide range of applications ranging from grid-scale battery storage scheduling to medical image segmentation. In the predict-then-optimize paradigm, the decision problem can be modeled by an

optimization problem with decision variables z (belonging to a well-defined and known feasible region $\mathcal{Z}$), an unknown parameter y, and a known objective function $f(y, z)$. Naturally, if y is perfectly known, we could directly solve the following optimization problem to obtain the optimal decision:

$$z_{\text{opt}} = \arg\min_{z \in \mathcal{Z}} f(y, z).$$

However, since we assume y is unknown, we may instead rely on a ML model to predict y from some observable contextual features $x \in \mathcal{X}$, where (x, y) share a joint distribution $\mathcal{P}$. One simple approach is to train a ML model $\hat{y}_\theta(x)$ to predict y from x, and then solve the optimization problem with the "plug-in" predicted value,

$$z_\theta^{\text{plug-in}}(x) = \arg\min_{z \in \mathcal{Z}} f(\hat{y}_\theta(x), z),$$

hence the name, "predict-then-optimize." More generally, besides the plug-in approach, we can explore other *policies*, denoted $z_\theta(x)$, that map the input x to a decision z. We will generally measure the performance of the policy by the expected task loss

$$F(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{P}}[f(y, z_\theta(x))],$$

which is the expected value of the objective function when the decision is made according to the policy $z_\theta(x)$ and the true parameter is y. We may also want our policy to satisfy certain reliability guarantees, such as controlling the tail risk of the task loss or ensuring robustness under distribution shift, expressed in the form $R(z_\theta) \leq \alpha$ for some function R and threshold $\alpha \in \mathbb{R}$. Thus, our overall objective is as follows:

$$\min_\theta F(\theta) \quad \text{subject to} \quad R(z_\theta) \leq \alpha. \tag{1.1}$$

Whereas the plug-in approach described above is a natural baseline, it does not necessarily optimize the task loss $F(\theta)$ or satisfy the reliability constraint $R(z_\theta) \leq \alpha$. Instead, Part I of this thesis develops methods that aim to more directly optimize the problem in (1.1).

The crucial difficulty is to both satisfy the reliability guarantee and learn good ML model parameters θ. At a high level, our approach is two-fold. First, we approach the constraint satisfaction problem by developing novel methods to accurately quantify the uncertainty of the ML model with theoretical guarantees, which we can then translate into constraint satisfaction guarantees. Second, we develop algorithms for end-to-end differentiation through the entire optimization problem (1.1) to enable

standard gradient-based optimization of the ML models, while preserving desired reliability guarantee. Each of the chapters in Part I explores different types of policies z_θ leveraging specific uncertainty quantification approaches with correspondingly different reliability guarantees.

Chapter 2 introduces conformal risk training, which generalizes conformal risk control from expected loss to optimized certainty equivalent risks, including CVaR, and then differentiates the risk-controlling procedure through the training loop. In this chapter, the ML model produces a point prediction whose risk is then controlled by a conformal procedure, but the point prediction is no longer treated as a fixed object. Instead, it is trained end-to-end with the risk-control parameter so that the model learns to place its uncertainty where it matters most for the downstream decision. This allows the model to become risk-aware: the learned predictor is shaped by how its uncertainty affects the final decision. Experiments on grid-scale battery storage optimization and a tumor image segmentation problem show that this can substantially improve decision quality while preserving the desired risk bound.

Chapter 3 moves from scalar risk control to set-valued uncertainty. It develops an end-to-end framework for conformally calibrated uncertainty sets in conditional robust optimization. The key idea is to learn the nonconformity score and the downstream decision jointly, so that the uncertainty set is calibrated and decision-aware at the same time. The chapter also shows that partially input-convex neural networks can represent a broad class of convex uncertainty sets while preserving tractable robust optimization. This yields a practical method for learning uncertainty sets that are expressive but still structured enough to optimize.

Chapter 4 takes the same decision-aware perspective but replaces prediction sets with learned conditional distributions and ambiguity sets. Rather than calibrating a set around a point predictor, it learns a conditional generative model and couples it to a Wasserstein distributionally robust optimization layer. In this setting, the ML model predicts a distribution over the uncertain parameter, and the downstream decision is formed by optimizing against an ambiguity set centered at that learned distribution. This makes it possible to propagate uncertainty through the decision problem in a way that is both flexible and tractable. The resulting Gen-WDRO method is evaluated on the grid-scale battery storage, where it improves robustness to distribution shift while remaining end-to-end differentiable.

Chapter 5 studies stochastic optimization with diffusion models. Standard DFL methods typically rely on deterministic point estimates, but many real problems are

inherently distributional. This chapter proposes a diffusion-based DFL framework that learns a full conditional distribution and optimizes the downstream decision under samples drawn from that distribution. In this case, the ML model directly predicts a distribution, and the downstream decision is optimized with respect to scenario samples from that distribution. The chapter develops both a reparameterization-based gradient estimator and a lightweight score-function estimator, the latter of which achieves comparable decision quality with dramatically lower memory usage. The chapter shows that diffusion models are a strong alternative to simpler parametric uncertainty models when the decision problem depends on the full shape of the distribution.

Part 2: online sequential decision-making

The second part of the thesis turns to online control problems with sequential decision-making. In these problems, the controller interacts with the environment repeatedly over time, and the environment may change in response to the controller's actions. This setting is more complex than the batch setting because the controller must learn while acting, and it must also account for how its actions affect future states of the environment. The chapters in Part II study two different types of sequential decision problems: an online voltage control problem with explicit stability guarantees, and a benchmark suite for sustainability-focused reinforcement learning (RL) that exposes realistic distribution shifts, physical constraints, and multi-agent interaction.

Chapter 6 studies voltage regulation on a distribution grid with unknown topology. The key technical challenge is that the controller must act before the topology is fully identified, and yet it must still keep the system stable. The chapter combines online nested convex body chasing with robust predictive control to obtain a finite-mistake guarantee. This is a strong form of safety in a sequential setting: the controller is allowed to make some errors while learning, but the total number of voltage violations is provably bounded.

Chapter 7 introduces SustainGym, a benchmark suite for reinforcement learning in sustainability applications. The environments model tasks such as EV charging, battery participation in electricity markets, carbon-aware datacenter scheduling, cogeneration, and building control. SustainGym is designed to expose realistic distribution shifts, physical constraints, and multi-agent interactions that are often missing from standard RL benchmarks. The chapter shows that off-the-shelf RL methods are brittle in these settings, which highlights the need for more robust

algorithms and more realistic evaluation environments.

Chapter 8 aims to address some of the shortcomings of multi-agent RL algorithms tested on SustainGym in Chapter 7 by developing a new family of distributionally robust multi-agent RL algorithms that can learn policies with strong out-of-distribution generalization guarantees. The key idea is to extend the individual-global-maximum (IGM) principle from value factorization to a distributionally robust setting, which yields a novel definition of robust individual action values that are compatible with decentralized greedy execution. The resulting DrIGM-compliant algorithms can be implemented as robust variants of existing value-factorization architectures (e.g., VDN/QMIX/QTRAN) without bespoke per-agent reward shaping. The chapter shows that these methods can achieve superior performance in SustainGym and game-playing environments.

1.3 Conclusion and Future Outlook

Across all of these chapters, the same design question reappears in different forms: what is the right representation of uncertainty for the decision at hand? Sometimes the answer is a calibrated threshold, sometimes a prediction set, sometimes a learned ambiguity set, and sometimes a sequential model of the system itself. The thesis argues that uncertainty should not be treated as an abstract statistical output. Uncertainty should be learned, shaped, and evaluated in terms of how it affects the final decision.

Taken together, the thesis develops a coherent perspective on learning decision-focused uncertainty representations. The common pattern is to make the uncertainty object itself part of the optimization problem, rather than a passive byproduct of prediction. This yields methods that are better aligned with the objectives of sustainability systems, and it suggests a general roadmap for future work: build uncertainty representations that are not only statistically meaningful, but also decision-aware, tractable, and robust enough for real-world deployment.

The final chapter of this thesis (Chapter 9) outlines some areas where the work of this thesis is readily extended, while also posing some open problems at the frontiers of decision-making under uncertainty.

Part I

End-to-End Learning for Uncertainty Representations

Chapter 2

CONFORMAL RISK TRAINING: END-TO-END OPTIMIZATION OF CONFORMAL RISK CONTROL

This chapter is primarily based on the following paper:

[306] Christopher Yeh, Nicolas Christianson, Adam Wierman, and Yisong Yue. 2025. Conformal Risk Training: End-to-End Optimization of Conformal Risk Control. In *Advances in Neural Information Processing Systems*. Vol. 38. San Diego, CA, USA, (Dec. 2025), 70056–70092. https://proceedings.neurips.cc/paper_files/paper/2025/hash/6559542f75b4452ebaaf82094c7defb7-Abstract-Conference.html.

2.1 Introduction

We study the problem of training deep learning models that are used for potentially risky downstream decision making. For example, in high-stakes tasks such as tumor classification, doctors need models that achieve both good overall classification accuracy and *provably* bounded false negative rate, to ensure that the health risk of a false negative prediction (i.e., misclassifying a tumor as benign) is sufficiently prioritized in model predictions. In such settings, it is important to design a unified approach (trained model and decision-making policy) that simultaneously controls for the desired level of risk while maximizing the utility of downstream decisions.

One promising paradigm is *risk control*. Given a (pre-trained) model whose predictions are used by a decision policy parameterized by $\lambda \in \Lambda$, the goal is to choose λ such that $\mathbb{E}[L(\lambda)] \leq \alpha$ for some loss function L and risk level α. Many common goals can be framed as risk control problems: bounding the false negative rate of a classifier, producing predictive uncertainty sets that satisfy a target coverage level, ensuring factuality of large language model outputs, etc. [11, 122].

The conformal risk control (CRC) method [11] introduces a sufficient criterion for solving the risk control problem when the loss function L is monotone. While CRC is elegant and simple, the original formulation has several limitations. First, CRC is limited to controlling the *expected* loss of L, while more general notions of risk are critical for high-stakes problems in the real world. Notably, the original paper on CRC [11] poses an open question on how to tackle more general notions of risk beyond expected loss, such as the conditional value-at-risk (CVaR). Second, because CRC is purely applied post-hoc (i.e., given a pre-trained model) to model outputs

without providing any feedback to the model, it may significantly degrade model performance.

In this chapter, we propose a theoretical and algorithmic framework, called *conformal risk training*, that extends CRC to enable end-to-end training and is applicable to tail risk formulations such as CVaR. Our key contributions are as follows:

1. First, we develop a risk control method for controlling the general class of *optimized certainty-equivalent (OCE)* risks [24, 23], a broad class of risk measures that includes as special cases the expected loss (generalizing the original CRC method) and the conditional value-at-risk (CVaR) [300]. In particular, the CVaR formulation partially answers an open question posed by the original CRC paper [11]. The key insight is that any OCE risk can be bounded by a monotone transformation of the loss, thereby preserving the monotonicity required for CRC-style methods.
2. Second, we propose *conformal risk training*, a method that trains a model and the conformal risk control procedure end-to-end. Our method substantially generalizes the method of conformal training of uncertainty sets [254, 311] to the conformal risk control setting. This trains the model to be "risk aware"—i.e., it learns to produce predictions that maximize performance while minimizing downstream risk.
3. Finally, we demonstrate empirically that fine-tuning models using conformal risk training leads to significant performance improvements while guaranteeing satisfaction of risk constraints. We present results for maximizing model specificity while controlling false negative rate on a tumor image segmentation task, as well as maximizing average profit while controlling tail-risk of losses in a battery storage operation task.

Related work. Previous works that explore post-hoc conformal risk control either bound the expectation of a monotone loss function, or provide a high-probability bound for (possibly) more general losses. We use the term "post-hoc" to refer to procedures that are applied to the outputs of a pretrained prediction model without further fine-tuning of the model. Within the class of bounds on expected losses, conformal prediction (CP) [278, 233, 9] bounds expected miscoverage loss for set-valued predictions, whereas conformal risk control (CRC) [11] generalizes CP to bound the expectation of any bounded monotone loss function. For high-probability risk bounds, Risk-Controlling Prediction Sets (RCPS) [21] bound the expected loss of set-valued predictors for monotone losses, whereas Learn Then Test (LTT) [10]

bounds more general risks and non-monotone losses at the cost of typically looser bounds from having to apply a family-wise error correction procedure. Recently, [52] developed a high-probability bound for the family of distortion risk measures which includes CVaR. While [11] shows how to convert a high-probability risk bound to an expected risk bound, it is not clear that their methodology can be extended to directly bound more general risks like CVaR. As such, to the best of our knowledge, our work is the first conformal-style approach that provides a *certain* (as opposed to high-probability) bound on risk measures beyond expected loss for monotone loss functions.

Several prior works within the CP literature have also explored calibrating model uncertainty during training. Conformal training [254] and related works [77, 59, 190] introduced methods for incorporating CP differentiably during model training by treating part of each minibatch as a pseudo-calibration set. These works primarily focus on reducing the size of calibrated prediction sets. In contrast, our *conformal risk training* method is the first to incorporate conformal risk control differentiably during model training and is compatible with more general performance objectives such as reducing the false positive rate of a classifier or minimizing an expected decision loss. We show in Section 2.10 that conformal training is a special case of our method.

Beyond the conformal prediction literature, a number of works in the machine learning literature (e.g., [146, 75, 152]) have introduced risk-sensitive objectives into learning; while these methods may, for example, reduce CVaR risk, they do not come with risk control guarantees.

Finally, our work is related to methods from "predict-then-optimize" [78] and decision-focused learning [175, 226], especially the growing literature on decision-focused uncertainty quantification (UQ). These methods aim to generate prediction sets that optimize some downstream decision-making objective while still maintaining calibration. Several of these methods are applied post-hoc to (potentially) decision-agnostic models [267, 60, 135], whereas others have combined conformal training with decision-focused learning [106, 311]. Our conformal risk training method builds upon this decision-focused UQ literature: whereas these prior works have largely focused on set-valued predictions and their associated risks, our method allows for more general risk measures.

Outline. This chapter is structured as follows. Section 2.2 introduces our overall problem setting and reviews the standard CRC result. Section 2.3 introduces our

broad generalization to the conformal control of OCE risks, of which standard CRC is a special case, and shows that this framework can be extended to more general assumptions in the specific case of CVaR risks. Section 2.4 introduces our conformal risk training procedure and discusses how the gradient of the risk-controlling parameter can be computed. Section 2.5 highlights key experimental results, and Section 2.6 concludes. Additional experimental results are presented in Section 2.7, and all proofs are deferred to Section 2.8.

Notation. $[N]$ denotes the set $\{1, 2, \ldots, N\}$. $[x]_+$ denotes the function $\max\{0, x\}$. $\mathbf{1}[\cdot]$ is the indicator function, while $\mathbf{1}_d$ is a length-d vector of ones. $\frac{\partial}{\partial x}$ denotes a partial derivative; $\frac{\mathrm{d}}{\mathrm{d}x}$ denotes the total derivative. $\partial f(x)$ is the subdifferential of a function f at a point x.

2.2 Preliminaries and Background on Conformal Risk Control

A primary goal in the training and deployment of machine learning (ML) models is to achieve both good overall performance and to ensure *reliable* deployment through the control of some notion of risk. To achieve this dual goal, existing approaches follow a two-stage, "Pretrain, then Risk Control" approach. First, an ML model with parameters $\theta \in \Theta \subseteq \mathbb{R}^D$ is trained to minimize a standard training objective, such as cross-entropy for classification. Then, after training, the decision-maker applies a post-hoc *risk control* procedure to the model to guarantee provably bounded risk. Formally, let $L : \Theta \times \Lambda \to \mathbb{R}$ denote a (random) mapping from model parameters $\theta \in \Theta$ and an "aggressiveness" parameter $\lambda \in \Lambda$ to some loss. The decision-maker seeks to choose the parameter λ to ensure that risk–the expectation of the loss L–is bounded at a chosen level α:

$$\mathbb{E}[L(\theta, \lambda)] \leq \alpha. \tag{2.1}$$

For instance, in a tumor image segmentation problem, L may denote the fraction of false negative pixels on a randomly drawn image, and α is a desired upper bound on the false negative rate (FNR). The aggressiveness parameter $\lambda \in [0, 1]$ can be chosen as the threshold used to distinguish positive predictions from negative ones. Thus, a smaller λ will yield more positive predictions and a lower false negative rate, and a greater λ will yield more negative predictions and a higher false negative rate. See Example 1 for a more detailed description of this task.

A number of post-hoc approaches to control the risk of pretrained machine learning models have been proposed in the literature. Most notable is the approach of *conformal risk control (CRC)* [11], which gives a distribution-free, finite-sample

methodology to provably enforce the risk bound (2.1). In the remainder of this section, we will omit the model parameters θ as an input to the loss function L, since the results hold beyond the case of controlling the risk of machine learning model decisions.

CRC assumes that the decision-maker has a dataset $\{L_1, \ldots, L_N\}$ of previous loss functions, and that the goal is to control the marginal loss of the next instance: $\mathbb{E}[L_{n+1}(\lambda)] \leq \alpha$. This can be done under several mild assumptions.

Assumption 1. *$\Lambda \subset \mathbb{R}$ is a closed and bounded set with minimum value $\lambda_{\min} := \min \Lambda$. The set $\{L_i : \Lambda \to \mathbb{R}\}_{i=1}^{N+1}$ is a set of exchangeable, left-continuous random functions. Furthermore, $B : \Lambda \to \mathbb{R}$ is a left-continuous function that almost surely upper bounds each L_i pointwise, so that for all $i \in [N+1]$, $\Pr(\forall \lambda \in \Lambda : \ L_i(\lambda) \leq B(\lambda)) = 1$.*

Assumption 2. *The functions $\{L_i\}_{i=1}^{N+1}$ are almost-surely nondecreasing, and B is nondecreasing.*

Assumption 3. *The risk control problem is* feasible*. That is, the desired risk level $\alpha \in \mathbb{R}$ is chosen such that $\mathbb{E}[L_{N+1}(\lambda_{\min})] \leq \alpha$.*

Under these assumptions, CRC gives the following approach for selecting λ to control risk.

Proposition 1. *Under Assumptions 1 and 2, let $\alpha \in \mathbb{R}$ be a desired risk level and define the set $\hat{\Lambda}$ as*

$$\hat{\Lambda} := \{\lambda \in \Lambda \mid h(\lambda) \leq \alpha\}, \quad \text{where} \quad h(\lambda) := \frac{1}{N+1}\left(B(\lambda) + \sum_{i=1}^{N} L_i(\lambda)\right). \tag{2.2}$$

Then, for any $\lambda \in \hat{\Lambda}$, we have $\mathbb{E}[L_{N+1}(\lambda)] \leq \alpha$. Furthermore, if Assumption 3 holds, then choosing $\hat{\lambda} := \max\{\lambda_{\min}, \sup \hat{\Lambda}\}$ ensures risk control: $\mathbb{E}[L_{N+1}(\hat{\lambda})] \leq \alpha$.

We make three brief remarks. First, the CRC procedure we describe above in Proposition 1 is actually a mild generalization of the original method in [11]; in particular, we allow the risk upper bound B to be a function of λ, whereas prior work assumes B to be a constant. In practice, this allows us to achieve tighter bounds on the risk. Second, we assume that the functions L_i are left-continuous and nondecreasing and we choose $\hat{\lambda}$ via the supremum, following the convention of [151]; on the other hand, the original CRC paper [11] assumes right-continuous and nonincreasing functions L_i and chooses a conservativeness parameter $\hat{\lambda}$ via the infimum. These

Algorithm 1 Conformal Risk Control (CRC)

Require: Parameter space $\Lambda = [\lambda_{\min}, \lambda_{\max}]$, loss functions $\{L_i : \Lambda \to \mathbb{R}\}_{i=1}^N$, upper-bound $B : \Lambda \to \mathbb{R}$, risk threshold $\alpha \in \mathbb{R}$, numerical tolerance $\epsilon > 0$

function CRC(Λ, $\{L_i\}_{i=1}^N$, B, α, ϵ)

  $\underline{\lambda} \leftarrow \lambda_{\min}$, $\overline{\lambda} \leftarrow \lambda_{\max}$

  while $\overline{\lambda} - \underline{\lambda} > \epsilon$ **do**

    $\lambda \leftarrow (\underline{\lambda} + \overline{\lambda})/2$

    if $\frac{1}{N+1}\big(B(\lambda) + \sum_{i=1}^N L_i(\lambda)\big) \leq \alpha$ **then** $\underline{\lambda} \leftarrow \lambda$ **else** $\overline{\lambda} \leftarrow \lambda$

  return $\underline{\lambda}$

approaches are equivalent up to reflection of the parameter λ. Finally, observe that due to monotonicity of the function $h(\lambda)$, the risk controlling parameter $\hat{\lambda}$ can be computed in practice through bisection search; see Algorithm 1 for a full description of this approach.

Standard CRC (Proposition 1) provides a sufficient criterion for choosing the parameter $\hat{\lambda}$ to ensure bounded *marginal* (*i.e.*, expected) loss. However, it does not address the control of more general notions of risk; in the next section, we will show that this framework can be broadly extended to the control of the general family of *optimized certainty equivalent* risks.

2.3 Conformal Risk Control for Optimized Certainty Equivalents

The previous section describes the control of *expected* loss to achieve the risk bound (2.1). However, in real-world, high-stakes applications, decision-makers may seek to control their risk beyond just the expected loss, especially if they are sensitive to losses of particular magnitudes. In general, they are faced with the problem of controlling their loss L under some chosen risk measure R, which maps from random variables to $\mathbb{R}$:

$$R[L(\lambda)] \leq \alpha. \tag{2.3}$$

While the CRC methodology described in the previous section can be extended to achieve the more general risk control (2.3) in certain special cases, such as when R is a quantile (see [11, Section 4.2]), there are many other important notions of risk which this approach cannot directly accommodate. For instance, in [11, Section 4.2], the authors pose the question of whether CRC or some other approach can be extended to enforce the risk control bound (2.3) when R is the *conditional value-at-risk* (CVaR), a common risk measure in financial and energy systems applications [220, 141, 184,

172].

In this section, we answer this question in the affirmative, proving that in fact, CRC can be extended to control any risk in the broad family of *optimized certainty equivalent* risks, defined as follows.

Definition 1 ([24, 23])**.** *A risk measure R mapping a real-valued random variable X to $\mathbb{R}$ is an* **optimized certainty equivalent (OCE)** *risk measure if $R[X]$ can be expressed as*

$$R[X] = \inf_{t\in\mathbb{R}} t + \mathbb{E}[\phi(X-t)],$$

where $\phi : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is a disutility function that is nondecreasing, closed, and convex with $\phi(0) = 0$ and $1 \in \partial\phi(0)$.

The family of OCE risks includes a number of practical and popular risk measures, including mean-variance and entropic risks [146]. Moreover, the *conditional value-at-risk* can also be shown to be an OCE risk measure with the disutility function $\phi_{\mathrm{CVaR}^\delta}(x) = \frac{1}{1-\delta}[x]_+$.

Definition 2 ([221, 220])**.** *Let X be a real-valued random variable, and let F be its cumulative distribution function. The* **conditional value-at-risk** *of X at level $\delta \in [0, 1)$, denoted $\mathrm{CVaR}^\delta[X]$, is the average value of X on its δ-tail, or the $1-\delta$ fraction of its largest outcomes. If X has a density, then $\mathrm{CVaR}^\delta[X] = \mathbb{E}[X \mid F(X) \geq \delta]$. For general random variables X, $\mathrm{CVaR}^\delta[X]$ can be expressed via the variational formula*

$$\mathrm{CVaR}^\delta[X] = \inf_{t\in\mathbb{R}} t + \frac{1}{1-\delta}\mathbb{E}[X-t]_+.$$

In Theorem 1, we show that the CRC methodology described in Proposition 1 can be broadly generalized to accommodate *any OCE risk measure*.

Assumption 4. *The OCE risk control problem is* feasible*. That is, the desired risk level α is chosen such that $R[L_{N+1}(\lambda_{\min})] \leq \alpha$.*

Theorem 1. *Suppose Assumptions 1 and 2 hold, and fix $\alpha, t \in \mathbb{R}$. Let R be an OCE risk measure with disutility function ϕ. Define the exchangeable random functions $\{\tilde{L}_{i,t} : \Lambda \to \mathbb{R}\}_{i=1}^{N+1}$ by $\tilde{L}_{i,t}(\lambda) := t + \phi(L_i(\lambda) - t)$, and let $\tilde{B}_t(\lambda) := t + \phi(B(\lambda) - t)$. Define*

$$\tilde{h}_t(\lambda) := \frac{1}{N+1}\left(\tilde{B}_t(\lambda) + \sum_{i=1}^{N}\tilde{L}_{i,t}(\lambda)\right), \tag{2.4}$$

and let $\hat{\Lambda}_t := \{\lambda \in \Lambda \mid \tilde{h}_t(\lambda) \leq \alpha\}$. *For every* $\lambda \in \hat{\Lambda}_t$, $R[L_{N+1}(\lambda)] \leq \alpha$. *Furthermore, if Assumption 4 holds, then choosing* $\hat{\lambda} := \max\{\lambda_{\min}, \sup \hat{\Lambda}_t\}$ *ensures risk control:* $R[L_{N+1}(\hat{\lambda})] \leq \alpha$.

The key insight underlying this theorem is that for any OCE risk, $\tilde{h}_t$ is nondecreasing in λ. This structure gives rise to the conformal OCE risk control (CORC) algorithm shown in Algorithm 2, which computes $\hat{\lambda}$ using bisection search.

Theorem 1 is a strict generalization of the original CRC methodology in Proposition 1. By choosing the disutility $\phi(x) = x$, the risk measure R is simply the expectation, and we recover the setting of controlling expected risk. In this special case, for all $t \in \mathbb{R}$, we have

$$\forall i \in [N] : \ \tilde{L}_{i,t} = L_i, \quad \tilde{B}_t = B, \quad \tilde{h}_t(\lambda) = h(\lambda), \quad \hat{\Lambda}_t = \hat{\Lambda},$$

thus recovering Proposition 1 exactly. Moreover, Theorem 1 answers positively the question from [11] of whether CRC can be generalized to control the CVaR, since the CVaR is an OCE risk measure. In fact, it is possible to obtain an even more general result in the specific case of controlling the CVaR. Specifically, we may relax the condition in Assumption 2 that the losses L_i are nondecreasing.

Assumption 5. *All* $\{L_i\}_{i=1}^{N+1}$ *and* B *are monotonic in* λ. *Note that within the set of functions* $\{B\} \cup \{L_i\}_{i=1}^{N+1}$, *we allow for some functions to be monotonically nondecreasing (e.g.,* L_1, L_3*), while others may be monotonically nonincreasing (e.g.,* L_2, B*).*

Even under this milder assumption on the structure of the losses L_i, we can obtain the following risk control guarantee for the CVaR.

Theorem 2. *Suppose that Assumptions 1 and 5 hold. Fix any* $\delta \in [0, 1)$ *and* $\alpha \in \mathbb{R}$. *Define* $\hat{\Lambda}_t$ *as in Theorem 1 using the disutility function* $\phi_{\mathrm{CVaR}^\delta}(x) = \frac{1}{1-\delta}[x]_+$. *Then, for every* $t \in [B(\lambda_{\min}), \alpha]$ *and* $\lambda \in \hat{\Lambda}_t$, *we have the risk control bound* $\mathrm{CVaR}^\delta\left[L_{N+1}(\lambda)\right] \leq \alpha$.

While the above result is written specifically for the case of CVaR, we note that the result can be extended to some, but not all, other OCE risk measures; see Section 2.8 for further details. Theorem 2 motivates the conformal CVaR control algorithm, which we present in Algorithm 2. Note that Theorem 2 is only non-vacuous

Algorithm 2 (Post-hoc) Conformal OCE Risk Control (CORC)

Require: parameter space $\Lambda = [\lambda_{\min}, \lambda_{\max}]$, functions $\{L_i\}_{i=1}^N$, upper-bound B, risk threshold $\alpha \in \mathbb{R}$, numerical tolerance $\epsilon > 0$, hyperparameter $t \in \mathbb{R}$

function CORC(Λ, $\{L_i\}_{i=1}^N$, B, α, ϵ, t, disutility $\phi : \mathbb{R} \to \mathbb{R}$)

$\underline{\lambda} \leftarrow \lambda_{\min}$, $\overline{\lambda} \leftarrow \lambda_{\max}$

while $\overline{\lambda} - \underline{\lambda} > \epsilon$ **do**

$\lambda \leftarrow (\underline{\lambda} + \overline{\lambda})/2$

if $\tilde{h}(\lambda, t) \leq \alpha$ **then** $\underline{\lambda} \leftarrow \lambda$ **else** $\overline{\lambda} \leftarrow \lambda$ ▷ *$\tilde{h}$ is defined in (2.4)*

return λ

function ConformalCVaRControl(Λ, $\{L_i\}_{i=1}^N$, B, α, ϵ, t, quantile level $\delta \in [0, 1)$)

if $\alpha < B(\lambda_{\min})$ or $t \notin [B(\lambda_{\min}), \alpha]$ **then return** $\lambda_{\min}$

return CORC(Λ, $\{L_i\}_{i=1}^N$, B, α, ϵ, t, $\phi_{\mathrm{CVaR}^\delta}$)

when $B(\lambda_{\min}) \leq \alpha$; otherwise, assuming feasibility of the risk control problem (Assumption 4), we can still use $\lambda_{\min}$ to obtain the desired bound.

Compared to the standard CRC result (Proposition 1), Theorems 1 and 2 both involve an additional hyperparameter t. For the risk bounds to hold, t should not depend on the calibration data $\{L_1, \ldots, L_N\}$ used to select the risk control parameter $\hat{\lambda}$. In practice, we recommend using an additional held-out set of losses $\{L'_1, \ldots, L'_k\}$ (e.g., the training set), and picking the t that yields the largest risk control parameter on this set.

While the conformal OCE and CVaR risk control procedures we have proposed in this section enable controlling substantially more general risks than the original CRC framework, such a post-hoc risk control procedure may still come with a substantial cost upon ML model deployment. For example, applying CRC to control the false negative rate of a model for tumor segmentation may come at the cost of a large false positive rate (see Figure 2.1 in our experiments). Because CRC is designed to be applied post-hoc to a pretrained model, there is no existing approach to remedy this degradation in performance. To improve performance while guaranteeing controlled risk, a better approach would *train* or *fine-tune* models subject to a constraint enforcing risk control. Designing a methodology to accomplish this goal is the focus of the next section.

Algorithm 3 Conformal Risk Training

Require: training set $\{(x_i, y_i)\}_{i=1}^M$, parameter space $\Lambda = [\lambda_{\min}, \lambda_{\max}]$, upper-bound $B : \Lambda \to \mathbb{R}$, disutility $\phi : \mathbb{R} \to \mathbb{R}$, risk threshold $\alpha \in \mathbb{R}$, initial model parameters θ

function ConformalRiskTraining($\{(x_i, y_i)\}_{i=1}^M$, Λ, B, ϕ, α, θ)

for mini-batch $D \subseteq [M]$ **do**

Randomly split batch: $(D_{\text{cal}}, D_{\text{pred}}) \leftarrow D$

Define loss functions $L_i(\theta, \lambda) := L(x_i, y_i, \theta, \lambda)$ for $i \in D_{\text{cal}}$

Define cost functions $\ell_i(\theta, \lambda) := \ell(x_i, y_i, \theta, \lambda)$ for $i \in D_{\text{pred}}$

With $\tilde{h}$ as defined in (2.4), compute,

$$\lambda(\theta) := \arg\min_{\lambda \in \Lambda} \min_{t \in [B(\lambda_{\min}), \alpha]} \frac{1}{|D_{\text{pred}}|} \sum_{i \in D_{\text{pred}}} \ell_i(\theta, \lambda) \quad \text{s.t. } \tilde{h}_t(\theta, \lambda) \leq \alpha$$

for $i \in D_{\text{pred}}$ **do**

Compute (sub)gradient of objective $\mathrm{d}\theta_i := \mathrm{d}\ell_i(\theta, \lambda(\theta))/\mathrm{d}\theta$

Update θ using (sub)gradients: $\sum_{i \in D_{\text{pred}}} \mathrm{d}\theta_i$

2.4 Conformal Risk Training

Returning to the setting described at the start of Section 2.2, the loss function L typically depends on the output of an ML model with parameters $\theta \in \Theta$. To explicitly denote this relationship, we write $L(\theta, \lambda)$, and we define $h(\theta, \lambda)$ (equation 2.2) and $\tilde{h}_t(\theta, \lambda)$ (equation 2.4) accordingly in terms of $\{L_i(\theta, \lambda)\}_{i=1}^N$. The CRC (Section 2.2) and CORC (Section 2.3) procedures for picking λ treat the model parameters θ as fixed. Thus, we call them "post-hoc" procedures.

However, as we show in our experiments, this separation of pre-training model parameters θ and performing risk control post-hoc leaves substantial room for improvement. Instead, in this section, we consider the problem of jointly optimizing the model parameters θ and the risk control parameter λ, which we call the *end-to-end optimal risk control problem*:

$$\min_{\theta \in \Theta,\, \lambda \in \Lambda} \mathbb{E}[\ell(\theta, \lambda)] \quad \text{s.t. } R[L(\theta, \lambda)] \leq \alpha, \tag{2.5}$$

where $\ell : \Theta \times \Lambda \to \mathbb{R}$ is a cost function that measures model performance and is differentiable almost everywhere. Note that we are specifically concerned with settings where the cost function ℓ depends on the risk control parameter λ. For a standard training objective such as cross-entropy where ℓ only depends on the model parameters θ but not on λ, the end-to-end optimal risk control problem (2.5) reduces

to standard empirical risk minimization.

The following example concretely illustrates the roles of ℓ and L for the problem of optimizing specificity in tumor image segmentation while controlling false negative rate.

Example 1. *In tumor image segmentation, an input image $X \in \mathcal{X} = \mathbb{R}^{d\times 3}$ (represented as a flattened array of RGB pixels) has a corresponding binary label $Y \in \mathcal{Y} = \{0,1\}^d$, where 1 indicates the presence of a tumor at a given pixel location. Let $f_\theta : \mathcal{X} \to [0,1]^d$ denote a segmentation model parameterized by θ that outputs a predicted probability that each pixel is tumorous. If $\lambda \in [0,1]$ is the decision threshold, let $\hat{Y}(\theta,\lambda) \in \mathcal{Y}$ denote the binarized prediction, i.e., for all $j \in [d]$, $\hat{Y}(\theta,\lambda)_j \coloneqq \mathbf{1}[f_\theta(X)_j \geq \lambda]$. Then, the fraction of false negative predictions is*

$$L(\theta,\lambda) \coloneqq 1 - \frac{|Y \wedge \hat{Y}(\theta,\lambda)|}{|Y|} = 1 - \frac{1}{|Y|}\sum_{j:\,Y_j=1} \mathbf{1}[f_\theta(X)_j \geq \lambda] \tag{2.6a}$$

$$= \frac{1}{|Y|}\sum_{j:\,Y_j=1} \mathbf{1}[f_\theta(X)_j < \lambda], \tag{2.6b}$$

where $|\cdot|$ counts the number of ones in a binary vector. Clearly, L is nondecreasing in λ. The expected loss $\mathbb{E}[L(\theta,\lambda)]$ gives the FNR of the model f_θ, and we can pick λ to control the FNR to be less than some target threshold α.

*However, in addition to controlling FNR, doctors and patients may also want to reduce false positives (i.e., optimize **specificity**). We can approximate the fraction of false positives in an image with*

$$\ell(\theta,\lambda) = \frac{1}{|\mathbf{1}_d - Y|}\sum_{j:\,Y_j=0} \sigma\left(\frac{f_\theta(X)_j - \lambda}{T}\right),$$

where T is a temperature hyperparameter.

As in the post-hoc CRC setting, we usually do not have access to the exact distribution of L (nor the distribution of ℓ). Instead, we only have a dataset of i.i.d. (or exchangeable) samples ℓ_i and L_i for $i \in [N]$, from which we can form the *end-to-end optimal CORC problem*:

$$\min_{\theta\in\Theta,\,\lambda\in\Lambda} \frac{1}{N}\sum_{i=1}^{N} \ell_i(\theta,\lambda) \quad \text{s.t. } \tilde{h}_t(\theta,\lambda) \leq \alpha. \tag{2.7}$$

We can equivalently express (2.7) as a bi-level optimization problem. The outer problem minimizes over θ, whereas the inner level chooses a risk-controlling λ:

$$\min_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} \ell_i(\theta, \lambda(\theta)), \quad \text{where } \lambda(\theta) := \arg\min_{\lambda\in\Lambda} \frac{1}{N}\sum_{i=1}^{N} \ell_i(\theta,\lambda) \text{ s.t. } \tilde{h}_t(\theta,\lambda) \le \alpha. \tag{2.8}$$

We shall assume that Assumption 4 holds, so in case the inner optimization problem in (2.8) is infeasible, we set $\lambda(\theta) = \lambda_{\min}$ to ensure risk control.

To solve (2.8) via gradient descent, we need to compute the gradient

$$\frac{\mathrm{d}\ell_i}{\mathrm{d}\theta}(\theta, \lambda(\theta)) = \frac{\partial \ell_i}{\partial \theta}(\theta, \lambda(\theta)) + \frac{\partial \ell_i}{\partial \lambda}(\theta, \lambda(\theta)) \cdot \frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta). \tag{2.9}$$

The key challenge lies in computing the derivative $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta)$. Fortunately, as we will show in Section 2.4, this derivative can be computed exactly in many common settings.

Assuming for now that we can compute $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta)$, we propose the *conformal risk training* method (Algorithm 3) for solving the general end-to-end CORC problem.[1] Inspired by the conformal training method [254], conformal risk training splits each minibatch of training data D into two halves, D_{cal} and D_{pred}. We use the pseudo-calibration set D_{cal} to form the loss functions L_i and compute the risk control parameter $\lambda(\theta)$, while we use D_{pred} to form the cost functions ℓ_i. Note that after training a model with conformal risk training, we use a fresh calibration dataset (and thus a fresh set of $L_1, \ldots, L_N$) to compute $\lambda(\theta)$ for use on a test input.

Computing the gradient in conformal risk training

This section discusses the computation of the gradient $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta)$. This is challenging in general, as it requires differentiating through the optimal solution of the lower-level CORC problem in (2.8), which may in general be nonconvex. Fortunately, it is possible to compute this gradient in many practical settings. In the following (informal) theorem, we give a high-level description of two cases in which we can compute this gradient; see Section 2.8 for a formal statement of the theorem and its proof.

Theorem (Informal version of Theorem 3, Section 2.8)**.** *Suppose Assumptions 1 and 2 hold and the inner problem in* (2.8) *is feasible. Under mild differentiability*

[1] The name "conformal risk training" was coined by David Stutz on his blog. However, to the best of our knowledge, we are the first to actually develop a concrete methodology for end-to-end conformal risk control.

conditions, we may obtain a closed-form expression for $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta)$ *in the following two cases:*

(i) if $\{\ell_i\}_{i=1}^N$ *are strictly decreasing in* λ*, and* $\{B\} \cup \{L_i\}_{i=1}^N$ *are piecewise constant in* λ*;*

(ii) if $\{\ell_i\}_{i=1}^N$ *are strictly convex or strictly monotone in* λ*, and* $\{B\} \cup \{L_i\}_{i=1}^N$ *are convex in* λ*.*

Note that when considering certain OCE risks like the CVaR in Theorem 3, the nondecreasing assumption (Assumption 2) can be relaxed to monotonicity (Assumption 5) in a similar manner as done in Theorem 2.

The gradient for conformal training [254] follows as a special case of (i) above (see Section 2.10). We now show that case (i) also applies immediately to the FNR loss considered in Example 1.

Example 2. *Recall the setting of Example 1. Given a dataset* $D_{cal} = \{(X_i, Y_i)\}_{i=1}^N$ *drawn exchangeably from* $\mathcal{P}$*, let* $L_i(\theta, \lambda) := 1 - |Y_i \wedge \hat{Y}_i(\theta, \lambda)|/|Y_i|$*. Clearly, every* L_i *is bounded by* $B = 1$*. Following the conformal risk control procedure, using the form (2.6) of the loss, and noting that each* ℓ_i *is strictly decreasing in* λ*, we may pick*

$$\lambda(\theta) := \max\left\{\lambda \in [0,1] \;\middle|\; \frac{1}{N+1}\left(1 + \sum_{i=1}^{N} \sum_{j:\,(Y_i)_j=1} \frac{1}{|Y_i|}\mathbf{1}[f_\theta(X_i)_j < \lambda]\right) \leq \alpha\right\}. \tag{2.10}$$

Assuming that all values $f_\theta(X_i)_j$ *are unique, then according to the proof of Theorem 3(i) (see Section 2.8), the optimal value* $\lambda(\theta) = f_\theta(X_i)_j$ *for some specific* (i, j)*. Thus, the gradient is*

$$\frac{\mathrm{d}\lambda(\theta)}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta} f_\theta(X_i)_j.$$

2.5 Experiments

In this section, we present experimental results for our conformal risk training method on two problems: (1) controlling false negative rate in tumor image segmentation [11], and (2) controlling CVaR of losses in grid-scale battery storage operation [73]. Code to reproduce our results is available on GitHub,[2] and additional experimental results and details are reported in Sections 2.7 and 2.9.

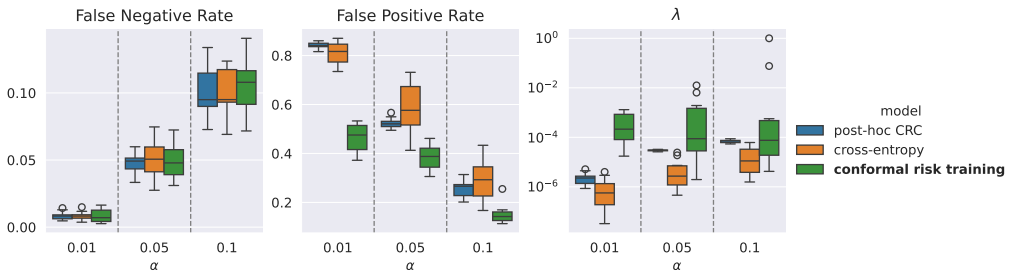


Figure 2.1: Results for the tumor image segmentation problem (Section 2.5) across different FNR thresholds α with 10 random seeds. **(left)** All three methods show FNR controlled at level α. **(middle)** Whereas only applying CRC post-hoc results in very large FPR, our method (conformal risk training, in green) is able to significantly reduce FPR without sacrificing FNR. **(right)** Our method generally picks a higher classification threshold λ than the baselines, suggesting that it is less conservative.

Controlling false negative rate in tumor image segmentation

We adopt the colonoscopy gut polyp image segmentation problem setup explored in [11, Section 3.1] and described in Example 1. We use a pre-trained PraNet [82] as our model f_θ, and we split images from 4 public datasets (CVC-ClinicDB [26], CVC-ColonDB [25], ETIS-LaribPolypDB [244], Kvasir-SEG [120]) into training, calibration, and test splits. In Figure 2.1, we compare the FNR and false positive rate (FPR) on the test set across three different models: (1) "post-hoc CRC" applied directly to the pre-trained PraNet; (2) "cross-entropy" refers to the fine-tuning PraNet using cross-entropy classification loss and then applying CRC; and (3) "conformal risk training" refers to fine-tuning PraNet using our method described in Section 2.4. For each model, we try 10 different random seeds for dividing the calibration and test splits, and we vary the target FNR α across three different values (0.01, 0.05, 0.1).

As Figure 2.1 shows, all three models have their expected FNR controlled at the target level α. However, for the "post-hoc CRC" and "cross-entropy" baselines, applying post-hoc CRC comes at a significant cost to the FPR, reaching as high as 80% FPR when the target FNR is 1%. In contrast, our conformal risk training method reduces FPR on average by 23-42% across the α levels. Furthermore, our method yields larger average values of λ than the baselines, suggesting that our method reduces conservativeness while maintaining the risk guarantee.

Controlling CVaR tail risk from battery storage operations

Energy resource operators seek electricity price forecasting models which deliver optimal expected profit while ensuring that the risk of losses due to poor predictions is

[2] https://github.com/chrisyeh96/conformal-risk-training

adequately controlled. We adopt a grid-scale battery operation problem [73], where a battery operator predicts electricity prices $Y \in \mathbb{R}^T$ over a T-step horizon and uses the predicted prices to decide how much to charge ($z^{\text{in}} \in \mathbb{R}^T$) or discharge ($z^{\text{out}} \in \mathbb{R}^T$) the battery, subject to constraints on the battery's state of charge expressed in terms of the net charge $z^{\text{net}} \in \mathbb{R}^T$. The input features X include the past day's prices and temperature, the current day's energy load forecast, and other date-related features. The battery has capacity C, charging efficiency γ, and maximum charge/discharge rates c^{in} and c^{out}. The task loss function f represents the multiple objectives of maximizing profit, battery health by discouraging rapid charging/discharging (with weight ϵ^{ramp}), and flexibility to participate in other markets by keeping the battery near half its capacity (with weight ϵ^{flex}):

$$f(y, z) = (z^{\text{in}} - z^{\text{out}})^\top y + \epsilon^{\text{ramp}} \left(\left\| z^{\text{in}} \right\|^2 + \left\| z^{\text{out}} \right\|^2 \right) + \epsilon^{\text{flex}} \left\| z^{\text{net}} \right\|^2,$$

with the following constraint set $\mathcal{Z}$ for $z := (z^{\text{in}}, z^{\text{out}}, z^{\text{net}}) \in \mathbb{R}^{3T}$:

$$\mathcal{Z} := \left\{ (z^{\text{in}}, z^{\text{out}}, z^{\text{net}}) \in \mathbb{R}^{3T} \;\middle|\; \begin{array}{c} 0 \le z^{\text{in}} \le c^{\text{in}}, \\ 0 \le z^{\text{out}} \le c^{\text{out}}, \\ -C/2 \le z^{\text{net}} \le C/2, \\ \forall t \in [T] : \; z_t^{\text{net}} := \sum_{\tau=1}^{t} \gamma z_\tau^{\text{in}} - z_\tau^{\text{out}} \end{array} \right\}. \tag{2.11}$$

Following [73], we set $T = 24$, $C = 1$, $\gamma = 0.9$, $c^{\text{in}} = 0.5$, $c^{\text{out}} = 0.2$, $\epsilon^{\text{flex}} = 0.1$, and $\epsilon^{\text{ramp}} = 0.05$.

If $z \in \mathcal{Z}$, then clearly $\lambda z \in \mathcal{Z}$ for all $\lambda \in \Lambda := [0, 1]$. We can thus implement a λ-dependent decision-maker via a "decision-scaling" rule $z_\theta(x, \lambda) = \lambda \hat{z}_\theta(x)$, where $\hat{z}_\theta(x) = \arg\min_{z \in \mathcal{Z}} f(\hat{y}_\theta(x), z)$ and $\hat{y}_\theta : \mathcal{X} \to \mathcal{Y}$ is a neural network pre-trained to minimize mean squared error in predicting electricity prices.[3] We use the task loss $\ell(\theta, \lambda) = f(Y, z_\theta(X, \lambda))$ as our training objective, which is strictly convex in λ, and we apply CVaR control to the financial risk term $L(\theta, \lambda) := \lambda(\hat{z}_\theta^{\text{in}}(X) - \hat{z}_\theta^{\text{out}}(X))^\top Y$. We construct a monotone-increasing upper bound $B(\lambda) := 100\lambda$ based on the empirical observation that our decision-maker $\hat{z}_\theta$ satisfies $(\hat{z}_\theta^{\text{in}}(x) - \hat{z}_\theta^{\text{out}}(x))^\top y \le 100$ on our training and validation sets. Note that Theorem 2 enables us to control the CVaR of $L(\theta, \lambda)$, which may either be monotone nondecreasing or nonincreasing in

[3] We do not train a model to directly map features x to decisions z because z must satisfy constraints. Instead, we train $\hat{y}_\theta$ to predict electricity prices, and then define the decision $\hat{z}_\theta(x)$ as a function of $\hat{y}_\theta(x)$.

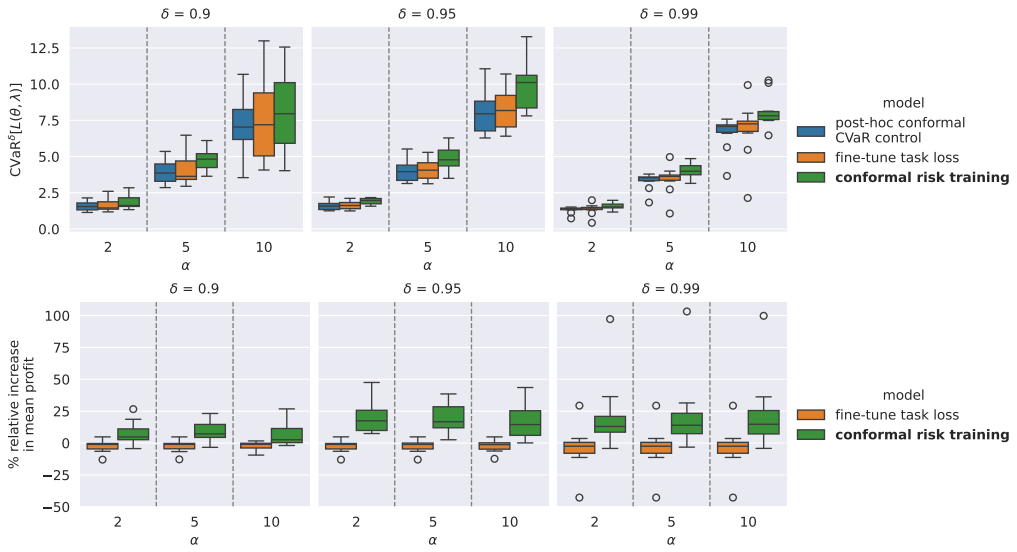


Figure 2.2: Results from the battery storage problem (Section 2.5) across different CVaR quantile levels δ and risk control thresholds α, with 10 random seeds. **(top)** All three methods show CVaR risk controlled at the target level α. **(bottom)** Comparison of the relative increase in profit (*i.e., negative task loss*) achieved by different methods over the post-hoc CRC baseline. Higher values are better.

λ; in contrast, the traditional CRC method (Proposition 1) does not apply, as it only allows for controlling the expectation of L when it is nondecreasing.

Because L and B are both convex in λ and differentiable almost everywhere in (θ, λ), $\lambda(\theta)$ is the solution to a convex optimization problem, and Theorem 3(ii) allows us to compute the derivative $\mathrm{d}\lambda(\theta)/\mathrm{d}\theta$ by differentiating through the KKT conditions of the convex optimization problem [3].

We compare test set tail risk and task loss across three different models: (1) "post-hoc conformal CVaR control" applied directly to the pretrained price prediction model $\hat{y}_\theta$; (2) "fine-tune task loss" refers to fine-tuning $\hat{y}_\theta$ using a decision-focused task-loss [73] and then applying conformal CVaR control; and (3) "conformal risk training" refers to fine-tuning $\hat{y}_\theta$ using the procedure described in Section 2.4. For each model, we try 10 different random seeds for dividing the validation and test splits, and we vary the target CVaR tail risk $\alpha \in (2, 5, 10)$ and the quantile level $\delta \in (0.9, 0.95, 0.99)$.

As Figure 2.2 (top) shows, all three models have their CVaR tail risk controlled at the target level α. However, in Figure 2.2 (bottom), it is clear that whereas fine-tuning using the task loss does not improve average task loss when controlling for tail risk, our conformal risk training method achieves between 7.2% and 22.6% mean improvement in task loss (*i.e.*, higher average profit) compared to the "post-hoc

Table 2.1: Sensitivity of task loss and tail risk to relative changes in hyperparameter t on the battery storage problem. Values given show mean ± 1 standard deviation of the task loss $\mathbb{E}_{(X,Y)\sim D}[f(Y, \lambda\hat{z}_\theta(X))]$ and financial tail risk $\mathrm{CVaR}^{\delta}_{(X,Y)\sim D}[\lambda(\hat{z}^{\mathrm{in}}_\theta(X) - \hat{z}^{\mathrm{out}}_\theta(X))^\top Y]$ for 10 models $\hat{y}_\theta$ trained with different random seeds. t is calculated as $t = (1 + \Delta t)t_0$, where t_0 denotes the optimal value of t tuned on the training set.

		$\alpha = 2$			$\alpha = 5$			$\alpha = 10$		
	Δt	$\delta = 0.9$	0.95	0.99	$\delta = 0.9$	0.95	0.99	$\delta = 0.9$	0.95	0.99
$\mathbb{E}[\ell]$	−0.5	−8.6 ± 3.2	−4.3 ± 1.6	−1.5 ± 0.5	−21.3 ± 8.0	−10.7 ± 4.0	−3.9 ± 1.2	−35.6 ± 6.3	−21.4 ± 7.9	−7.7 ± 2.5
	−0.1	−8.6 ± 3.2	−4.3 ± 1.6	−1.8 ± 0.5	−21.3 ± 7.9	−10.7 ± 4.0	−4.4 ± 1.3	−35.6 ± 6.3	−21.5 ± 7.9	−8.9 ± 2.6
	0	−8.6 ± 3.2	−4.3 ± 1.6	−1.8 ± 0.5	−21.3 ± 7.9	−10.7 ± 4.0	−4.5 ± 1.3	−35.6 ± 6.3	−21.5 ± 7.9	−9.0 ± 2.6
	0.1	−8.6 ± 3.2	−4.3 ± 1.6	−1.8 ± 0.5	−21.3 ± 7.9	−10.7 ± 4.0	−4.6 ± 1.2	−35.6 ± 6.3	−21.5 ± 7.9	−9.1 ± 2.5
	0.5	−8.6 ± 3.2	−4.3 ± 1.6	−1.7 ± 0.4	−21.3 ± 7.9	−10.7 ± 4.0	−4.1 ± 1.0	−35.6 ± 6.3	−21.5 ± 7.9	−8.3 ± 2.0
$\mathrm{CVaR}^{\delta}[L]$	−0.5	1.6 ± 0.3	1.6 ± 0.3	1.1 ± 0.2	3.9 ± 0.8	4.0 ± 0.8	2.8 ± 0.6	7.1 ± 2.2	8.0 ± 1.5	5.6 ± 1.1
	−0.1	1.6 ± 0.3	1.6 ± 0.3	1.3 ± 0.2	3.9 ± 0.8	4.0 ± 0.8	3.3 ± 0.6	7.0 ± 2.2	8.1 ± 1.5	6.5 ± 1.2
	0	1.6 ± 0.3	1.6 ± 0.3	1.3 ± 0.2	3.9 ± 0.8	4.0 ± 0.8	3.3 ± 0.6	7.0 ± 2.2	8.1 ± 1.5	6.6 ± 1.2
	0.1	1.6 ± 0.3	1.6 ± 0.3	1.3 ± 0.2	3.9 ± 0.8	4.0 ± 0.8	3.4 ± 0.6	7.0 ± 2.2	8.1 ± 1.5	6.7 ± 1.2
	0.5	1.6 ± 0.3	1.6 ± 0.3	1.2 ± 0.3	3.9 ± 0.8	4.0 ± 0.8	3.1 ± 0.7	7.0 ± 2.2	8.1 ± 1.5	6.2 ± 1.5
t_0	-	0.0 ± 0.0	0.0 ± 0.0	0.8 ± 0.3	0.0 ± 0.0	0.0 ± 0.0	2.0 ± 0.7	0.1 ± 0.1	0.1 ± 0.1	4.1 ± 1.5

conformal CVaR control" baseline at all tested risk thresholds α and quantile levels δ.

Sensitivity to t hyperparameter. As noted in Section 2.3, conformal CVaR control requires picking a hyperparameter $t \in [B(\lambda_{\min}), \alpha]$, which we set to the value t_0 that yields the largest risk control parameter $\hat{\lambda}$ on the training set. To understand the sensitivity of conformal CVaR control to the choice of t, we computed task loss and the empirical CVaR of financial risk under both relative (Table 2.1) and absolute (Table 2.3) perturbations of t away from t_0. Empirically, we find that the task loss and CVaR are not very sensitive to changes in t; the task loss from perturbed t tends to be close to (within 1 standard deviation of) the task loss from t_0.

2.6 Conclusion

We have developed the conformal OCE risk control method, which is a strict generalization of the original CRC procedure. In particular, this allows us to directly control the CVaR tail risk, unlike previous works which only control expected losses or provide high-probability bounds. We have also developed conformal risk training, which generalizes the conformal training procedure from conformal prediction to the setting of conformal OCE risk control, and our experiments show significant improvements in model performance over applying post-hoc CRC alone.

Limitations and future directions. The main limitations of conformal OCE risk control are the same limitations that apply to standard CRC: the risk control guarantee only applies to monotone and exchangeable losses. For conformal risk training, while

Table 2.2: Comparison of cross-entropy loss, false negative rate (FNR), and false positive rate (FPR) across baseline methods ("post-hoc CRC" and "cross-entropy") and our conformal risk training (CRT) method. The α values in the different columns indicate the level at which FNR is controlled by CRC. The α value in parenthesis following "CRT" indicates the FNR threshold enforced during conformal risk training.

	cross-entropy loss	$\alpha = 0.01$		$\alpha = 0.05$		$\alpha = 0.1$	
		FNR	FPR	FNR	FPR	FNR	FPR
post-hoc CRC	2.8 ± 0.1	0.009 ± 0.003	0.841 ± 0.014	0.048 ± 0.008	0.524 ± 0.022	0.101 ± 0.019	0.256 ± 0.035
cross-entropy	2.9 ± 0.2	0.008 ± 0.003	0.811 ± 0.049	0.050 ± 0.014	0.580 ± 0.106	0.101 ± 0.017	0.293 ± 0.092
CRT (α = .01)	11.9 ± 0.4	0.008 ± 0.005	0.465 ± 0.057	-	-	-	-
CRT (α = .05)	5.8 ± 2.5	-	-	0.049 ± 0.014	0.385 ± 0.054	-	-
CRT (α = .1)	7.8 ± 6.1	-	-	-	-	0.106 ± 0.021	0.152 ± 0.041

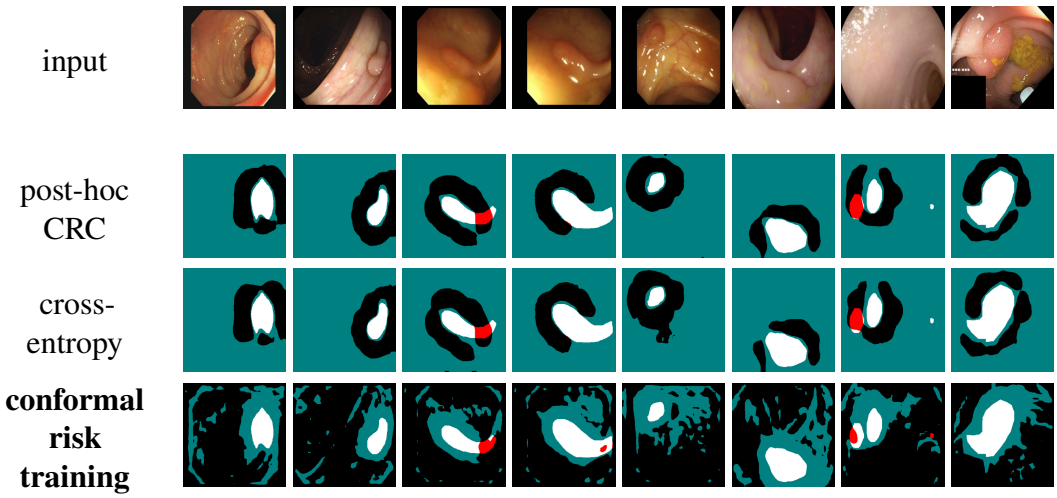
we derive the exact gradient in some common cases, we do not provide a complete characterization of when the gradient exists.

Future work may examine the tightness of the conformal OCE risk control bound and consider generalizations of CRC to other families of risk measures such as distortion [52] or coherent risk measures [15]. We believe that conformal OCE risk control will be of particular interest to high-stakes applications in finance, robotics, and LLM alignment, where provable tail risk guarantees are critical.

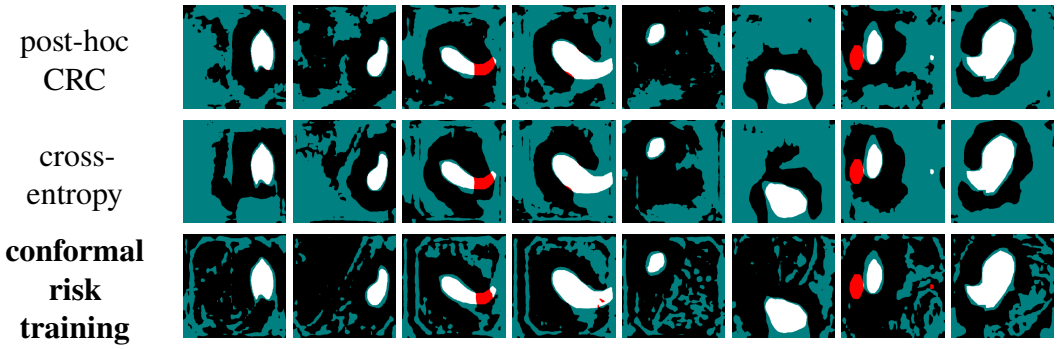
2.7 Additional Experimental Results

Tumor image segmentation. Figure 2.3 shows a random selection of 8 images from the test set, along with the corresponding predictions made by the baseline methods ("post-hoc CRC" and "cross-entropy") compared to our "conformal risk training" method. Evidently, our method significantly reduces false positives (shown in teal) without significantly increasing false negatives (shown in red), especially when the false negative rate is controlled at a very low α. Note that the marginal risk control guarantee offered by the CRC procedure (Algorithm 1) ensures that false negative rate will be controlled at the target level *on average* over test examples, while individual examples might exceed the target false negative rate. This is why in certain cases, our method ("conformal risk training") might exceed the false negative rate of other methods (e.g., in the 7th column of the $\alpha = 0.05$ case in Figure 2.3), while on average, it guarantees the target false negative rate.

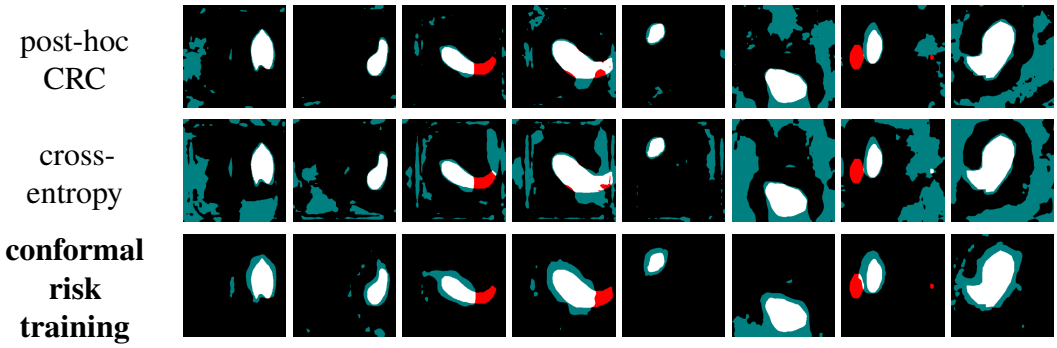
We would also like to emphasize the decision-focused nature of our method. As shown in Figure 2.1, further training the pre-trained image segmentation model with the PraNet weighted cross-entropy loss [82] and then applying post-hoc CRC (shown in orange) does not improve model performance over directly applying post-hoc CRC



(a) $\alpha = 0.01$



(b) $\alpha = 0.05$



(c) $\alpha = 0.1$

Figure 2.3: Predictions on 8 randomly selected images from the test set for the tumor image segmentation problem (Section 2.5). Black and white pixels indicate correct outputs. False positives are shaded teal; false negatives are shaded red.

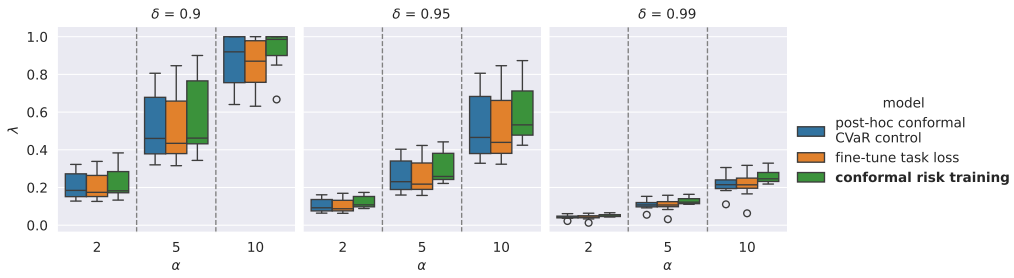


Figure 2.4: Values of the decision scaling factor λ for the battery storage problem (Section 2.5) across different CVaR quantile levels δ and risk control thresholds α, with 10 random seeds. Higher values indicate more aggressiveness and larger charge/discharge decisions.

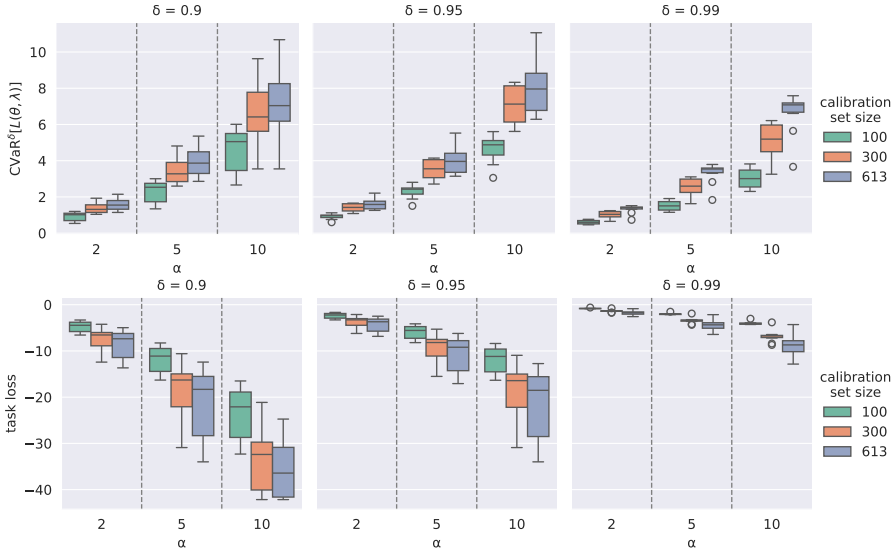


Figure 2.5: Comparison of financial tail risk (top) and task loss (bottom) as a function of calibration set size on the battery storage problem (Section 2.5), across different CVaR quantile levels δ and risk control thresholds α, with 10 random seeds. Here, post-hoc conformal CVaR control is applied to the pre-trained prediction model $\hat{y}_\theta$.

to the pre-trained model (shown in blue). This observation is verified in Table 2.2, where we see no meaningful difference in performance between the "post-hoc CRC" and "cross-entropy" rows. In contrast, at a given false negative rate (FNR) risk level α, our conformal risk training method achieves lower false positive rate (FPR) but incurs higher cross-entropy loss.

Battery storage operations. In Figure 2.4, we show values of the decision scaling factor λ obtained through our conformal risk training approach compared with the baselines described in Section 2.5. In general across values of the CVaR quantile level δ and risk control threshold α, it appears that our conformal risk training

Table 2.3: Sensitivity of task loss and tail risk to absolute changes in hyperparameter t on the battery storage problem described in Section 2.5. Values given show mean ± 1 stddev of the task loss $\mathbb{E}_{(X,Y)\sim D}[f(Y, \lambda\hat{z}_\theta(X))]$ and the financial tail risk $\text{CVaR}^{\delta}_{(X,Y)\sim D}[\lambda(\hat{z}^{\text{in}}_\theta(X) - \hat{z}^{\text{out}}_\theta(X))^\top Y]$ for 10 models $\hat{y}_\theta$ trained with different random seeds. t is calculated as $t = t_0 + \Delta t$, where t_0 denotes the optimal value of t tuned on the training set. In the case that $t_0 + \Delta t \notin [B(\lambda_{\min}), \alpha]$, no values are given, as indicated by "-".

		$\alpha = 2$			$\alpha = 5$			$\alpha = 10$		
	Δt	$\delta = 0.9$	0.95	0.99	$\delta = 0.9$	0.95	0.99	$\delta = 0.9$	0.95	0.99
$\mathbb{E}[\ell]$	−5	-	-	-	-	-	-	-	-	−6.5 ± 0.9
	−2	-	-	-	-	-	−3.3 ± 1.2	-	-	−8.2 ± 2.5
	−1	-	-	−1.3 ± 0.2	-	-	−4.1 ± 1.2	-	-	−9.1 ± 2.3
	0	−8.6 ± 3.2	−4.3 ± 1.6	−1.8 ± 0.5	−21.3 ± 7.9	−10.7 ± 4.0	−4.5 ± 1.3	−35.6 ± 6.3	−21.5 ± 7.9	−9.0 ± 2.6
	1	−6.8 ± 1.9	−4.4 ± 1.1	−1.0 ± 0.7	−19.9 ± 6.7	−11.3 ± 3.6	−4.5 ± 0.8	−35.4 ± 6.3	−22.2 ± 7.5	−9.3 ± 2.1
	2	−0.7 ± 0.3	−0.8 ± 0.0	0.0 ± 0.0	−18.1 ± 5.5	−11.3 ± 3.1	−3.4 ± 1.3	−34.8 ± 6.2	−22.6 ± 7.0	−8.9 ± 1.5
	5	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	−0.4 ± 0.9	−1.9 ± 0.7	0.0 ± 0.0	−32.2 ± 7.2	−21.9 ± 5.6	−4.8 ± 3.5
$\text{CVaR}^{\delta}[L]$	−5	-	-	-	-	-	-	-	-	4.4 ± 0.3
	−2	-	-	-	-	-	2.3 ± 0.6	-	-	5.9 ± 1.0
	−1	-	-	0.9 ± 0.1	-	-	3.0 ± 0.5	-	-	6.6 ± 0.7
	0	1.6 ± 0.3	1.6 ± 0.3	1.3 ± 0.2	3.9 ± 0.8	4.0 ± 0.8	3.3 ± 0.6	7.0 ± 2.2	8.1 ± 1.5	6.6 ± 1.2
	1	1.3 ± 0.3	1.7 ± 0.2	0.8 ± 0.6	3.7 ± 0.8	4.3 ± 0.6	3.4 ± 0.5	7.0 ± 2.1	8.4 ± 1.4	6.9 ± 1.0
	2	0.1 ± 0.1	0.3 ± 0.1	0.0 ± 0.0	3.4 ± 0.8	4.3 ± 0.6	2.6 ± 1.2	6.9 ± 2.1	8.6 ± 1.3	6.7 ± 1.1
	5	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.1	0.7 ± 0.3	0.0 ± 0.0	6.2 ± 1.7	8.4 ± 1.0	3.8 ± 3.0
t_0	-	0.0 ± 0.0	0.0 ± 0.0	0.8 ± 0.3	0.0 ± 0.0	0.0 ± 0.0	2.0 ± 0.7	0.1 ± 0.1	0.1 ± 0.1	4.1 ± 1.5

method yields values of λ that are larger on average than the alternative methods. This implies that our method is able to learn to deploy *more aggressive* decisions, and employs a greater portion of the battery capacity when charging/discharging than the alternative methods. In other words, by learning to use a larger scaling factor λ while preserving the risk constraint, the electricity price forecasting model trained via conformal risk training becomes more "calibrated" in regard to risk. In addition, as the value of δ increases, the scaling factor λ decreases across all methods, reflecting the increased need to make more conservative decisions in settings where losses on the extreme tail must be controlled.

Conformal CVaR control performs better when the calibration set is larger. As shown in Figure 2.5, a larger calibration set generally reduces the task loss and achieves a tighter CVaR bound closer to α (i.e., not being overly conservative). The reason a larger calibration set size N helps is because the term $\frac{1}{N+1}\tilde{B}_t(\lambda)$ appears in the expression for $\tilde{h}_t(\lambda)$. Increasing N reduces the effect of the upper bound B and enables choosing larger (less conservative) λ. However, there are diminishing gains as N increases.

In Table 2.3, we compute the task loss and CVaR financial tail risk on the test set under perturbations of the hyperparameter t from the tuned value t_0. Because Theorem 2 only holds when $t \in [B(\lambda_{\min}), \alpha]$, we exclude reporting results when the

perturbed t is outside of the required interval. We find that while our t_0 tuned based on the training set is not always optimal for the test set, it is generally close to the optimal.

2.8 Deferred Proofs

Throughout the proofs, we sometimes rely on the property that a disutility function $\phi : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ corresponding to an OCE risk measure (Definition 1) is a closed function. For completeness, we provide the definition of a closed function below.

Definition 3 ([28]). *A function $f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is* **closed** *if its epigraph* $\operatorname{epi} f := \{(x,t) \in \mathbb{R}^2 \mid f(x) \le t\}$ *is a closed set in* $\mathbb{R}^2$.

Proof of Proposition 1

Define the set

$$\hat{\Lambda}' := \{\lambda \in \Lambda \mid \hat{R}(\lambda) \le \alpha\}, \quad \text{where} \quad \hat{R}(\lambda) := \frac{1}{N+1} \sum_{i=1}^{N+1} L_i(\lambda),$$

and recall the definition of the set $\hat{\Lambda}$ from (2.2), which we state again below:

$$\hat{\Lambda} := \{\lambda \in \Lambda \mid h(\lambda) \le \alpha\}, \quad \text{where} \quad h(\lambda) := \frac{1}{N+1} \left(B(\lambda) + \sum_{i=1}^{N} L_i(\lambda) \right).$$

Since $L_i \le B$ almost surely for every $i \in [N]$, $\hat{R} \le h$ almost surely. Therefore, $h(\lambda) \le \alpha$ implies $\hat{R}(\lambda) \le \alpha$ almost surely, so $\hat{\Lambda} \subseteq \hat{\Lambda}'$ almost surely.

Define $\bar{\lambda} := \sup \hat{\Lambda}$ and $\bar{\lambda}' := \sup \hat{\Lambda}'$. We follow the convention that $\sup \emptyset = -\infty$. Since $\hat{\Lambda} \subseteq \hat{\Lambda}'$ almost surely, we have $\bar{\lambda} \le \bar{\lambda}'$ almost surely.

We now consider two cases based on whether $\hat{\Lambda}'$ is empty or not.

1. Suppose $\hat{\Lambda}' = \emptyset$. Then $\hat{\Lambda} = \emptyset$ almost surely, so it is (vacuously) true that for all $\lambda \in \hat{\Lambda}$, $\mathbb{E}[L_{N+1}(\lambda)] \le \alpha$.

   If Assumption 3 holds, then $\bar{\lambda} = \bar{\lambda}' = -\infty$ almost surely, so $\hat{\lambda} = \max\{\lambda_{\min}, \bar{\lambda}\} = \lambda_{\min}$ almost surely. Therefore, $\mathbb{E}[L_{N+1}(\hat{\lambda})] \le \alpha$.

2. Suppose $\hat{\Lambda}'$ is non-empty. In this case, $\bar{\lambda}' = \max \hat{\Lambda}'$ because (1) $\{L_i\}_{i=1}^{N+1}$ are left-continuous functions, so $\hat{R}$ is also left-continuous; (2) $\hat{\Lambda}'$ contains at least one point; and (3) we assume that Λ is closed.

Let E be the random multiset of loss functions $\{L_1, \ldots, L_{N+1}\}$, and observe that $\bar{\lambda}'$ is a constant conditional on E. The random functions L_i are exchangeable, which implies that

$$L_{N+1}(\bar{\lambda}') \mid E \;\sim\; \text{Uniform}\left\{L_1(\bar{\lambda}'), \ldots, L_{N+1}(\bar{\lambda}')\right\}.$$

Therefore,

$$\mathbb{E}\left[L_{N+1}(\bar{\lambda}') \mid E\right] = \frac{1}{N+1}\sum_{i=1}^{N+1} L_i(\bar{\lambda}') = \hat{R}(\bar{\lambda}').$$

Note that $\hat{R}$ is a random function, because the L_i's are random. By the law of total expectation (i.e., by further taking an expectation over the random multiset E), we have

$$\mathbb{E}\left[L_{N+1}(\bar{\lambda}')\right] = \mathbb{E}\left[\hat{R}(\bar{\lambda}')\right] \leq \alpha$$

where the expectation is over randomness from E, and the inequality comes from the definitions of $\hat{\Lambda}'$ and $\bar{\lambda}'$.

Since L_{N+1} is nondecreasing almost surely and $\bar{\lambda} \leq \bar{\lambda}'$ almost surely, we have

$$\mathbb{E}\left[L_{N+1}(\bar{\lambda})\right] \leq \mathbb{E}\left[L_{N+1}(\bar{\lambda}')\right] \leq \alpha.$$

Since $\bar{\lambda} = \sup \hat{\Lambda}$ and L_{N+1} is nondecreasing almost surely,

$$\forall \lambda \in \hat{\Lambda}: \quad \mathbb{E}[L_{N+1}(\lambda)] \leq \mathbb{E}[L_{N+1}(\bar{\lambda})] \leq \alpha.$$

If Assumption 3 holds, then observe that both

$$\mathbb{E}[L_{N+1}(\bar{\lambda})] \leq \alpha \quad \text{and} \quad \mathbb{E}[L_{N+1}(\lambda_{\min})] \leq \alpha.$$

Therefore, $\mathbb{E}[L_{N+1}(\hat{\lambda})] \leq \alpha$.

□

Proof of Theorem 1

Lemma 1. *If $\phi : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is a disutility function corresponding to an optimized certainty equivalent risk (Definition 1), then ϕ is left-continuous.*

Furthermore, if ϕ only takes on real values, then ϕ is continuous.

Proof of Lemma 1. From the definition of an optimized certainty equivalent risk (Definition 1), $\phi : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is a closed, convex, nondecreasing function. By [28, Proposition 1.1.2], closedness implies ϕ is lower-semicontinuous.

Next, we show that since ϕ is lower-semicontinuous and nondecreasing, it is left-continuous. Fix any $x_0 \in \mathbb{R}$. Since ϕ is monotone, it has a left-limit at x_0, which we shall call L:

$$L := \lim_{x \uparrow x_0} \phi(x).$$

Since ϕ is nondecreasing, $L \leq \phi(x_0)$. Choose any increasing sequence $\{x_n \in \mathbb{R}\}_{n \in \mathbb{N}}$ that converges to x_0. Then by definition of left-limit, we have

$$\lim_{n \to \infty} \phi(x_n) \to L.$$

Lower semicontinuity at x_0 yields

$$\phi(x_0) \leq \liminf_{x \to x_0} \phi(x) \leq \liminf_{n \to \infty} \phi(x_n) = \lim_{n \to \infty} \phi(x_n) = L.$$

We have thus shown $L \leq \phi(x_0) \leq L$, so $\phi(x_0) = L$, so ϕ is left-continuous at x_0. Since x_0 was arbitrary, ϕ is left-continuous on all of $\mathbb{R}$.

For the case where ϕ is real-valued (i.e., is never infinite), then it is a well-known fact that convex real-valued functions are continuous. □

Proof of Theorem 1. Fix any $t \in \mathbb{R}$. By Lemma 1, ϕ is left-continuous. Since $(\phi, \{L_i\}_{i=1}^{N+1}, B)$ are nondecreasing left-continuous functions, and the composition of nondecreasing left-continuous functions remains nondecreasing left-continuous, $(\{\tilde{L}_{i,t}\}_{i=1}^{N+1}, \tilde{B}_t)$ are nondecreasing left-continuous functions. Furthermore, we have $\tilde{B}_t(\lambda) \geq \tilde{L}_{i,t}(\lambda)$ almost surely.

For any $\hat{\lambda} \in \hat{\Lambda}_t$, we have

$$\begin{aligned} R[L_{N+1}(\hat{\lambda})] &= \inf_{\hat{t} \in \mathbb{R}} \mathbb{E}\left[\hat{t} + \phi(L_{N+1}(\hat{\lambda}) - \hat{t})\right] \\ &\leq \mathbb{E}\left[t + \phi(L_{N+1}(\hat{\lambda}) - t)\right] = \mathbb{E}[\tilde{L}_{N+1,t}(\hat{\lambda})] \\ &\leq \alpha, \end{aligned}$$

where the last inequality comes from Proposition 1. □

Proof of Theorem 2

Theorem 2 states that CVaR risk can be controlled when the loss functions L_i are monotone, as opposed to the stronger nondecreasing requirement from Proposition 1.

In this section, we prove the more general result (Proposition 2) that when L_i are monotone, any OCE risk measure whose disutility function can be written as the composition of another OCE risk measure's disutility ϕ_{base} and $[.]_+$ can also be controlled. We then show that Theorem 2 follows as corollary of this more general result.

Lemma 2. *If $f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is a closed function (Definition 3), and $g : \mathbb{R} \to \mathbb{R}$ is continuous, then $f \circ g$ is closed.*

Proof of Lemma 2. Let (x_n, t_n) be a sequence of points in $\text{epi}(f \circ g)$ such that $(x_n, t_n) \to (x, t)$. By definition of epigraph, $f(g(x_n)) \leq t_n$, so $(g(x_n), t_n) \in \text{epi}\, f$. Since g is continuous, $g(x_n) \to g(x)$, and thus $(g(x_n), t_n) \to (g(x), t)$. Since f is closed, $\text{epi}\, f$ is closed, so $(g(x), t) \in \text{epi}\, f$. In other words, $f(g(x)) \leq t$, so $(x, t) \in \text{epi}(f \circ g)$.

Thus, we have shown that every convergent sequence of points in $\text{epi}(f \circ g)$ converges within $\text{epi}(f \circ g)$, so $\text{epi}(f \circ g)$ is closed. Therefore, $f \circ g$ is closed. □

Proposition 2. *Suppose that Assumptions 1 and 5 hold. Fix any $\alpha \in \mathbb{R}$, and let $\phi_{\text{base}} : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ be a disutility function for some OCE risk measure. Let R be the OCE risk measure corresponding to the transformed disutility function $\phi(x) := \phi_{\text{base}}([x]_+)$, and define $\hat{\Lambda}_t$ as in Theorem 1 using this transformed disutility function ϕ. Then, for every $t \in [B(\lambda_{\min}), \alpha]$ and every $\lambda \in \hat{\Lambda}_t$, we have the risk control bound $R[L_{N+1}(\lambda)] \leq \alpha$.*

Proof of Proposition 2. First, we show that ϕ is a valid disutility function. By definition of a disutility function, ϕ_{base} is convex, closed, and nondecreasing. ϕ is convex because it is the composition of a convex, nondecreasing function ϕ_{base} and a convex function $[\cdot]_+$. ϕ is nondecreasing because it is the composition of two nondecreasing functions. By Lemma 2, ϕ is closed because it is the composition of a closed function ϕ_{base} and a continuous function $[\cdot]_+$. Furthermore, $\phi(0) = \phi_{\text{base}}([0]_+) = 0$. Finally, to show that $1 \in \partial\phi(0)$, we consider two cases. If $x < 0$, then $\phi(x) - \phi(0) = \phi(0) - \phi(0) = 0 > x$. If $x \geq 0$, then $\phi(x) - \phi(0) = \phi_{\text{base}}(x) \geq x$ since $1 \in \partial\phi_{\text{base}}(0)$. Thus, for all $x \in \mathbb{R}$, $\phi(x) - \phi(0) \geq x$, which shows that $1 \in \partial\phi(0)$. Therefore, ϕ is a valid disutility function.

Fix any $t \geq B(\lambda_{\min})$. Following Theorem 1, we define

$$\tilde{L}_{i,t}(\lambda) := t + \phi(L_i(\lambda) - t) \qquad \forall i \in [N+1]$$
$$\tilde{B}_t(\lambda) := t + \phi(B(\lambda) - t).$$

We will establish that $\tilde{L}_{i,t}$ is nondecreasing and left-continuous. By Assumption 5, L_i is monotone in λ. If L_i is nondecreasing in λ, then clearly $[L_i(\lambda)-t]_+$ is nondecreasing in λ as well. If L_i is decreasing in λ, observe that $L_i(\lambda_{\min}) \leq B(\lambda_{\min}) \leq t$ almost surely, so $[L_i(\lambda) - t]_+ = 0$ almost surely, for all $\lambda \in \Lambda$. Therefore, $[L_i(\lambda) - t]_+$ is almost surely nondecreasing in λ. By Assumption 1, L_i is left-continuous, so $[L_i(\lambda)-t]_+$ is likewise left-continuous in λ. ϕ_{base} is also nondecreasing (by definition of disutility function) and left-continuous (by Lemma 1). The composition of two nondecreasing left-continuous functions remains nondecreasing and left-continuous, so $\phi(L_i(\lambda) - t) = \phi_{\text{base}}([L_i(\lambda) - t]_+)$ is nondecreasing and left-continuous in λ. Therefore, $\tilde{L}_{i,t}$ is nondecreasing and left-continuous.

By a similar argument, $\tilde{B}_t$ is also nondecreasing and left-continuous. Furthermore, since $B(\lambda) \geq L_i(\lambda)$ almost surely, and ϕ is nondecreasing, we have $\tilde{B}_t(\lambda) \geq \tilde{L}_{i,t}(\lambda)$ almost surely.

Thus, $\tilde{B}_t$ and $\{\tilde{L}_{i,t}\}_{i=1}^{N+1}$ satisfy Assumptions 1 and 2. Therefore, by Proposition 1, for every $\hat{\lambda} \in \hat{\Lambda}_t$,

$$\begin{aligned} R[L_{N+1}(\hat{\lambda})] &= \inf_{\hat{t} \in \mathbb{R}} \mathbb{E}\left[\hat{t} + \phi\left(L_{N+1}(\hat{\lambda}) - \hat{t}\right)\right] \\ &\leq \mathbb{E}\left[t + \phi\left(L_{N+1}(\hat{\lambda}) - t\right)\right] = \mathbb{E}\left[\tilde{L}_{N+1,t}(\hat{\lambda})\right] \\ &\leq \alpha. \end{aligned}$$

We remark that if $t \leq \alpha$ (which requires $\alpha \geq B(\lambda_{\min})$), then $\hat{\Lambda}_t$ is almost surely nonempty. Note that $\tilde{B}_t(\lambda_{\min}) = t \leq \alpha$. Every $\tilde{L}_{i,t}$ is almost surely bounded above by $\tilde{B}_t$, so $\tilde{L}_{i,t}(\lambda_{\min}) \leq \alpha$ almost surely. Therefore, $\lambda_{\min} \in \hat{\Lambda}_t$ almost surely.

On the other hand, if $t > \alpha$, then $\hat{\Lambda}_t$ is almost surely empty. This is because for every $\lambda \in \Lambda$, we have

$$\tilde{B}_t(\lambda) \geq \tilde{L}_{i,t}(\lambda) \geq \tilde{L}_{i,t}(\lambda_{\min}) \geq t > \alpha \quad \text{almost surely,}$$

so $\tilde{h}_t(\lambda) = \frac{1}{N+1}\left(\tilde{B}_t(\lambda) + \sum_{i=1}^{N} \tilde{L}_{i,t}(\lambda)\right) > \alpha$ almost surely. Therefore, for $t > \alpha$, Proposition 2 is almost surely vacuously true.

□

Proof of Theorem 2. Theorem 2 follows as a corollary of Proposition 2, where we set $\phi_{\text{base}} = \phi_{\text{CVaR}^\delta}$. Then, $\phi = \phi_{\text{CVaR}^\delta}$:

$$\phi(x) := \phi_{\text{base}}([x]_+) = \phi_{\text{CVaR}^\delta}([x]_+) = \frac{1}{1-\delta}[[x]_+]_+ = \frac{1}{1-\delta}[x]_+ = \phi_{\text{CVaR}^\delta}(x).$$

□

Theorem 3 and its Proof

We start by stating and proving several technical lemmas that will help in the proof of Theorem 3. Throughout this section, we abbreviate "almost everywhere" as "a.e."

Lemma 3. *If $L : \Lambda \to \mathbb{R}$ is convex and $\phi : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is convex and nondecreasing, then the function $\tilde{L} : \Lambda \times \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ defined by $\tilde{L}(\lambda, t) := t + \phi(L(\lambda) - t)$ is convex in (λ, t).*

Proof. Let $a \in [0, 1]$, and fix any $(\lambda_1, t_1), (\lambda_2, t_2) \in \Lambda \times \mathbb{R}$. Suppose L is convex and ϕ is convex and nondecreasing. Then,

$$\begin{aligned}
&\tilde{L}(a\lambda_1 + (1-a)\lambda_2, at_1 + (1-a)t_2) \\
&= at_1 + (1-a)t_2 + \phi(L(a\lambda_1 + (1-a)\lambda_2) - (at_1 + (1-a)t_2)) \\
&\leq at_1 + (1-a)t_2 + \phi(aL(\lambda_1) + (1-a)L(\lambda_2) - (at_1 + (1-a)t_2)) \\
&= at_1 + (1-a)t_2 + \phi(a(L(\lambda_1) - t_1) + (1-a)(L(\lambda_2) - t_2)) \\
&\leq at_1 + (1-a)t_2 + a\phi(L(\lambda_1) - t_1) + (1-a)\phi(L(\lambda_2) - t_2) \\
&= a\tilde{L}(\lambda_1, t_1) + (1-a)\tilde{L}(\lambda_2, t_2).
\end{aligned}$$

□

Lemma 4. *Suppose $L : \Theta \times \Lambda \to \mathbb{R}$ is a 0-1 step function in λ of the form*

$$L(\theta, \lambda) = \mathbf{1}[g(\theta) < \lambda]$$

where $g : \Theta \to \mathbb{R}$ is a continuous function whose level sets have measure zero. Then, L is continuous a.e. in θ.

Proof. Fix $\lambda \in \mathbb{R}$, and define the preimages $U = g^{-1}((-\infty, \lambda))$, $V = g^{-1}((\lambda, \infty))$, and $W = g^{-1}(\{\lambda\})$. Clearly, $\Theta = U \cup V \cup W$. Since the level sets of g have measure zero, W has measure 0.

Since g is continuous, U and V are open sets. For any $\theta \in U$, $L(\theta, \lambda) = 1$ is constant, so L is continuous on U. Likewise, L is constant and hence continuous on V.

Thus, L is continuous in θ everywhere except on W, a set of measure 0. □

Lemma 5. *Suppose $L : \Theta \times \Lambda \to \mathbb{R}$ is piecewise constant, left-continuous, and nondecreasing in λ with the form*

$$L(\theta, \lambda) = c_0(\theta) + \sum_{j \in \mathbb{N}} c_j(\theta) \cdot \mathbf{1}[g_j(\theta) < \lambda]$$

where $(c_j : \Theta \to \mathbb{R})_{j \in \mathbb{Z}_+}$ and $(g_j : \Theta \to \mathbb{R})_{j \in \mathbb{N}}$ are continuous functions, $(c_j(\theta))_{j \in \mathbb{N}}$ are nonnegative, and $(g_j(\theta))_{j \in \mathbb{N}}$ is nondecreasing. Assume that the level sets of $(g_j)_{j \in \mathbb{N}}$ have measure zero.

Let $\phi : \mathbb{R} \to \mathbb{R}$ be an OCE disutility function, and let $t \in \mathbb{R}$. Then, the function $\tilde{L} : \Theta \times \Lambda \to \mathbb{R}$ defined as

$$\tilde{L}(\theta, \lambda) := t + \phi(L(\theta, \lambda) - t) = t + \phi\left(c_0(\theta) - t + \sum_{j \in \mathbb{N}} c_j(\theta) \cdot \mathbf{1}[g_j(\theta) < \lambda]\right)$$

is continuous a.e. in θ, piecewise-constant left-continuous nondecreasing in λ, and can be written in the same form as L, i.e.,

$$\tilde{L}(\theta, \lambda) = \tilde{c}_0(\theta) + \sum_{j \in \mathbb{N}} \tilde{c}_j(\theta) \cdot \mathbf{1}[g_j(\theta) < \lambda]$$

where $(\tilde{c}_j)_{j \in \mathbb{Z}_+}$ are continuous functions and $(\tilde{c}_j(\theta))_{j \in \mathbb{N}}$ are nonnegative.

Proof. For all $k \in \mathbb{N}$, define

$$c'_k(\theta) := t + \phi\left(c_0(\theta) - t + \sum_{j=1}^{k} c_j(\theta)\right).$$

Since ϕ is a nondecreasing function and $(c_j(\theta))_{j \in \mathbb{N}}$ are nonnegative, $(c'_j(\theta))_{j \in \mathbb{Z}_+}$ is a nondecreasing sequence of reals. $(c_j)_{j \in \mathbb{Z}_+}$ are continuous by assumption, and ϕ is continuous by Lemma 1, so $(c'_j)_{j \in \mathbb{Z}_+}$ are continuous functions.

Observe that $\tilde{L}$ is also piecewise-constant left-continuous nondecreasing in λ:

$$\tilde{L}(\theta, \lambda) = \begin{cases} c'_0(\theta), & \lambda \in (-\infty, g_1(\theta)] \\ c'_k(\theta), & \lambda \in (g_k(\theta), g_{k+1}(\theta)]. \end{cases}$$

Thus, we can write

$$\tilde{L}(\theta, \lambda) = \tilde{c}_0(\theta) + \sum_{j \in \mathbb{N}} \tilde{c}_j(\theta) \cdot \mathbf{1}[g_j(\theta) < \lambda],$$

where

$$\tilde{c}_0(\theta) = c'_0(\theta), \qquad \tilde{c}_j(\theta) := c'_j(\theta) - c'_{j-1}(\theta) \quad \forall j \in \mathbb{N}.$$

Because $(c'_j(\theta))_{j\in\mathbb{Z}_+}$ are nondecreasing, $(\tilde{c}_j(\theta))_{j\in\mathbb{N}}$ are nonnegative. Furthermore, $(\tilde{c}_j)_{j\in\mathbb{Z}_+}$ inherit continuity from $(c'_j)_{j\in\mathbb{Z}_+}$.

By Lemma 4, each $\mathbf{1}[g_j(\theta) < \lambda]$ is continuous a.e. in θ. The product of continuous a.e. functions is continuous a.e., so each term $\tilde{c}_j(\theta) \cdot \mathbf{1}[g_j(\theta) < \lambda]$ is continuous a.e. in θ. The countable sum of continuous a.e. functions remains continuous a.e., so $\tilde{L}$ is continuous a.e. in θ. □

Now, we are ready to state the formal assumptions needed for Theorem 3.

Assumption 6. *The functions B and $\{L_i\}_{i=1}^N$ are of the form*

$$B(\lambda) = b_0 + \sum_{j\in\mathbb{N}} b_j \cdot \mathbf{1}[d_j < \lambda], \qquad L_i(\theta,\lambda) = c_{i,0}(\theta) + \sum_{j\in\mathbb{N}} c_{i,j}(\theta) \cdot \mathbf{1}[g_{i,j}(\theta) < \lambda]$$

where

1. $(c_{i,j} : \Theta \to \mathbb{R})_{i\in[N],j\in\mathbb{Z}_+}$ *and* $(g_{i,j} : \Theta \to \mathbb{R})_{i\in[N],j\in\mathbb{N}}$ *are continuous functions;*
2. $(b_j \in \mathbb{R})_{j\in\mathbb{N}}$ *and* $(c_{i,j}(\theta))_{i\in[N],j\in\mathbb{N}}$ *are nonnegative;*
3. $(d_j \in \mathbb{R})_{j\in\mathbb{N}}$ *and* $(g_{i,j}(\theta))_{j\in\mathbb{N}}$ *for all i are nondecreasing sequences, and* $\{d_j\}_{j\in\mathbb{N}} \cup \{g_{i,j}(\theta)\}_{i\in[N],j\in\mathbb{N}}$ *are all unique; and*
4. $(g_{i,j})_{i\in[N],j\in\mathbb{N}}$ *are differentiable a.e., and all of their level sets have measure zero.*

Assumption 7. *Suppose that the inner problem in (2.8) exhibits strong duality (e.g., Slater's condition holds). Let $\mu(\theta)$ denote the optimal Lagrange multiplier arising from the dual problem to (2.8). Define*

$$\tilde{\ell}(\theta,\lambda) := \begin{cases} \lambda, & \text{if } \sum_{i=1}^N \ell_i \text{ is strictly increasing in } \lambda \\ -\lambda, & \text{if } \sum_{i=1}^N \ell_i \text{ is strictly decreasing in } \lambda \\ \frac{1}{N}\sum_{i=1}^N \ell_i(\theta,\lambda), & \text{if } \sum_{i=1}^N \ell_i \text{ is strictly convex in } \lambda \end{cases}.$$

The following regularity conditions hold:

1. $\frac{\partial}{\partial\lambda}\tilde{\ell}$ *exists and is continuously differentiable in (θ,λ) in a neighborhood around $(\theta,\lambda(\theta))$;*
2. $B \cup \{L_i\}_{i=1}^N$ *are twice continuously differentiable in (θ,λ) in a neighborhood around $(\theta,\lambda(\theta))$;*

3. *ϕ is twice continuously differentiable in neighborhoods around $B(\lambda(\theta)) - t$ and $L_i(\theta, \lambda(\theta)) - t$ for each $i \in [N]$; and*
4. $\begin{bmatrix} \frac{\partial^2}{\partial \lambda^2}\tilde{\ell}(\theta, \lambda(\theta)) + \mu(\theta) \cdot \frac{\partial^2}{\partial \lambda^2}\tilde{h}_t(\theta, \lambda(\theta)) & \frac{\partial \tilde{h}_t}{\partial \lambda}(\theta, \lambda(\theta)) \\ \mu(\theta) \cdot \frac{\partial \tilde{h}_t}{\partial \lambda}(\theta, \lambda(\theta)) & \tilde{h}_t(\theta, \lambda(\theta)) - \alpha \end{bmatrix}$ *is invertible.*

With these assumptions stated, we now state the full version of Theorem 3.

Theorem 3. *Let $\phi : \mathbb{R} \to \mathbb{R}$ be an OCE disutility function. Suppose Assumptions 1 and 2 hold and that the inner problem in (2.8) is feasible.*

(i) Suppose $\sum_{i=1}^N \ell_i$ is strictly decreasing in λ; the set $\{\theta \in \Theta \mid \tilde{h}_t(\theta, \lambda) = \alpha\}$ has measure zero for all λ; and $\{B\} \cup \{L_i\}_{i=1}^N$ are piecewise constant in λ. Under the standard differentiability and regularity conditions in Assumption 6, we may obtain a closed-form expression for $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}$ a.e.

(ii) Suppose $\sum_{i=1}^N \ell_i$ is strictly convex or strictly monotone in λ, and $\{B\} \cup \{L_i\}_{i=1}^N$ are convex in λ. Under the standard differentiability and regularity conditions in Assumption 7, the derivative $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta)$ exists and has a closed-form expression.

Proof. **Setting (i).** By Lemma 5, each $\tilde{L}_{i,t}(\theta, \lambda) := t + \phi(L_i(\theta, \lambda) - t)$ is piecewise-constant left-continuous nondecreasing in λ and can be written in the form

$$\tilde{L}_{i,t}(\theta, \lambda) = \tilde{c}_{i,0}(\theta) + \sum_{j \in \mathbb{N}} \tilde{c}_{i,j}(\theta) \cdot \mathbf{1}[g_{i,j}(\theta) < \lambda],$$

where $(\tilde{c}_{i,j})_{j \in \mathbb{Z}_+}$ are continuous functions and $(\tilde{c}_{i,j}(\theta))_{j \in \mathbb{N}}$ are nonnegative. Similarly, $\tilde{B}_t$ is piecewise-constant left-continuous nondecreasing and can be written in the form

$$\tilde{B}_t(\lambda) = \tilde{b}_0 + \sum_{j \in \mathbb{N}} \tilde{b}_j \cdot \mathbf{1}[d_j < \lambda]$$

where $(\tilde{b}_j)_{j \in \mathbb{N}}$ are nonnegative. Thus, $\tilde{h}_t(\theta, \lambda) := \frac{1}{N+1}\left(\tilde{B}_t(\lambda) + \sum_{i=1}^N \tilde{L}_{i,t}(\theta, \lambda)\right)$ is piecewise-constant left-continuous nondecreasing in λ and can be written in the form

$$\tilde{h}_t(\theta, \lambda) = c_0^{(h)}(\theta) + \sum_{j \in \mathbb{N}} c_j^{(h)}(\theta) \cdot \mathbf{1}[g_j^{(h)}(\theta) < \lambda].$$

The sequence of steps is given by

$$(g_j^{(h)}(\theta))_{j \in \mathbb{N}} = \mathrm{sort\_ascending}(\{d_j\}_{j \in \mathbb{N}} \cup \{g_{i,j}(\theta)\}_{i \in [N], j \in \mathbb{N}}),$$

which yields a strictly increasing sequence because $\{d_j\}_{j \in \mathbb{N}} \cup \{g_{i,j}(\theta)\}_{i \in [N], j \in \mathbb{N}}$ are all unique by assumption. The nonnegative coefficients $(c_j^{(h)}(\theta))_{j \in \mathbb{Z}_+}$ are defined

appropriately in terms of $(\tilde{b}_j)_{j\in\mathbb{Z}_+}$ and $(\tilde{c}_{i,j})_{i\in[N],j\in\mathbb{Z}_+}$, and $(c_j^{(h)})_{j\in\mathbb{Z}_+}$ inherit their continuity.

The functions $(g_j^{(h)})_{j\in\mathbb{N}}$ are differentiable a.e. and continuous; these properties are inherited from $(d_j)_{j\in\mathbb{N}}$ (which are constant) and $(g_{i,j})_{i\in[N],j\in\mathbb{N}}$ (which are differentiable a.e. and continuous by assumption).

By Lemma 5, each $\tilde{L}_{i,t}$ is continuous a.e. in θ. $\tilde{B}_t$ is constant in θ, and therefore also continuous in θ. $\tilde{h}_t$ is a countable, weighted sum of $\tilde{B}_t$ and $\tilde{L}_{i,t}$ for $i \in [N]$, so it is continuous a.e. in θ.

Recall from (2.8) that

$$\lambda(\theta) := \arg\min_{\lambda\in\Lambda} \frac{1}{N}\sum_{i=1}^{N} \ell_i(\theta,\lambda) \;\text{ s.t. } \tilde{h}_t(\theta,\lambda) \le \alpha.$$

Because each ℓ_i is strictly decreasing in λ, we may equivalently write the optimization problem as maximizing λ subject to the constraint on $\tilde{h}_t$ and leave the optimal solution $\lambda(\theta)$ unaffected:

$$\lambda(\theta) = \max_{\lambda\in\Lambda} \lambda \;\text{ s.t. } \tilde{h}_t(\theta,\lambda) \le \alpha.$$

By assumption, the optimization problem is feasible and $\tilde{h}_t(\theta,\lambda) \neq \alpha$ θ-almost everywhere. Thus, for all θ except on a negligible set, one of the two following cases must hold:

1. $\tilde{h}_t(\theta,\lambda) < \alpha$ for every $\lambda \in \Lambda$.

   Then, $\lambda(\theta) = \max\Lambda$. We showed above that $\tilde{h}_t(\theta, \max\Lambda)$ is continuous a.e. in θ. Thus, for θ a.e., there exists a neighborhood $\mathcal{N}(\theta)$ around θ for which $\tilde{h}_t(\theta', \max\Lambda) < \alpha$ for all $\theta' \in \mathcal{N}(\theta)$. Thus, $\lambda(\theta) = \max\Lambda$ for all $\theta' \in \mathcal{N}(\theta)$, so $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta) = \frac{\mathrm{d}}{\mathrm{d}\theta}\max\Lambda = 0$.

2. There exists an index $k \in \mathbb{N}$ such that

$$c_0^{(h)}(\theta) + \sum_{j=1}^{k} c_j^{(h)}(\theta)\cdot \mathbf{1}[g_j^{(h)}(\theta) < \lambda(\theta)] < \alpha,$$

$$c_0^{(h)}(\theta) + \sum_{j=1}^{k+1} c_j^{(h)}(\theta)\cdot \mathbf{1}[g_j^{(h)}(\theta) < \lambda(\theta)] > \alpha.$$

   In this case, because the objective seeks to maximize λ and $(g_j^{(h)}(\theta))_{j\in\mathbb{N}}$ is a strictly increasing sequence, $\lambda(\theta) = g_{k+1}^{(h)}(\theta)$.

As shown above, $g_{k+1}^{(h)}$ is differentiable a.e. Now, suppose that $g_{k+1}^{(h)}$ is differentiable at θ. Consider any entry θ_q of the parameter vector θ, and let e_q denote the standard unit basis vector along coordinate q. Then the partial derivative of λ with respect to θ_q is

$$\begin{aligned}\frac{\partial \lambda}{\partial \theta_q}(\theta) &= \lim_{s\to 0} \frac{\lambda(\theta + se_q) - \lambda(\theta)}{s} \\ &= \lim_{s\to 0} \frac{\lambda(\theta + se_q) - g_{k+1}^{(h)}(\theta)}{s} \\ &= \lim_{s\to 0} \frac{g_{k+1}^{(h)}(\theta + se_q) - g_{k+1}^{(h)}(\theta)}{s} \\ &= \frac{\partial g_{k+1}^{(h)}}{\partial \theta_q}(\theta).\end{aligned}$$

The third equality comes from observing that $(c_j^{(h)})_{j\in\mathbb{Z}_+}$ and $(g_j^{(h)})_{j\in\mathbb{N}}$ are continuous functions, so there exists an $\epsilon > 0$ such that for all $s \in [0, \epsilon]$,

$$c_0^{(h)}(\theta + se_q) + \sum_{j=1}^{k} c_j^{(h)}(\theta + se_q) \cdot \mathbf{1}[g_j^{(h)}(\theta + se_q) < \lambda(\theta + se_q)] < \alpha$$

$$c_0^{(h)}(\theta + se_q) + \sum_{j=1}^{k+1} c_j^{(h)}(\theta + se_q) \cdot \mathbf{1}[g_j^{(h)}(\theta + se_q) < \lambda(\theta + se_q)] > \alpha.$$

Thus, for all $s \in [0, \epsilon]$, $\lambda(\theta + se_q) = g_{k+1}^{(h)}(\theta + se_q)$.

Since $\frac{\partial \lambda}{\partial \theta_q} = \frac{\partial g_{k+1}^{(h)}}{\partial \theta_q}$ for all coordinates q, we have $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta) = \frac{\mathrm{d}g_{k+1}^{(h)}}{\mathrm{d}\theta}(\theta)$.

Therefore, we have derived a closed-form expression for $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta)$ almost everywhere.

Setting (ii). Since ϕ is convex and nondecreasing, $\{\tilde{B}_t\} \cup \{\tilde{L}_{i,t}\}_{i=1}^N$ are convex in λ by Lemma 3. Therefore, $\tilde{h}_t$ is convex in λ.

Recall from (2.8) that

$$\lambda(\theta) := \arg\min_{\lambda\in\Lambda} \frac{1}{N} \sum_{i=1}^{N} \ell_i(\theta, \lambda) \;\text{ s.t. } \tilde{h}_t(\theta, \lambda) \le \alpha.$$

If $\sum_{i=1}^N \ell_i$ is strictly increasing (or decreasing) in λ but not strictly convex in λ, observe that we may replace the objective $\frac{1}{N}\sum_{i=1}^N \ell_i$ with simply λ (or $-\lambda$) in (2.8) without changing the solution $\lambda(\theta)$. Concretely, let

$$\tilde{\ell}(\theta, \lambda) := \begin{cases} \lambda, & \text{if } \sum_{i=1}^N \ell_i \text{ is strictly increasing in } \lambda \\ -\lambda, & \text{if } \sum_{i=1}^N \ell_i \text{ is strictly decreasing in } \lambda \\ \frac{1}{N}\sum_{i=1}^N \ell_i(\theta, \lambda), & \text{if } \sum_{i=1}^N \ell_i \text{ is strictly convex in } \lambda \end{cases}$$

Then,

$$\lambda(\theta) = \arg\min_{\lambda \in \Lambda} \tilde{\ell}(\theta, \lambda) \;\text{ s.t. }\; \tilde{h}_t(\theta, \lambda) \leq \alpha. \tag{2.12}$$

Since $\lambda(\theta)$ is the solution to a convex optimization problem whose objective $\tilde{\ell}$ is either strictly convex or convex and strictly monotone, the solution is unique. Every convex optimization problem can be equivalently reformulated as a convex conic problem [3]. Then, the implicit function theorem, applied to the KKT equality conditions, gives sufficient conditions for $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta)$ to exist [7, 4, 3].

Specifically, define the Lagrangian $\mathcal{L} : \Lambda \times \mathbb{R} \times \Theta \to \mathbb{R}$ and the function $g : \Lambda \times \mathbb{R} \times \Theta \to \mathbb{R}^2$ by

$$\mathcal{L}(\lambda, \mu, \theta) := \tilde{\ell}(\theta, \lambda) + \mu \cdot (\tilde{h}_t(\theta, \lambda) - \alpha)$$
$$g(\lambda, \mu, \theta) := \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \lambda}(\lambda, \mu, \theta) \\ \mu \cdot (\tilde{h}_t(\theta, \lambda) - \alpha) \end{bmatrix}.$$

The KKT equality conditions for primal-dual optimality of $(\lambda(\theta), \mu(\theta))$ are precisely given by

$$g(\lambda(\theta), \mu(\theta), \theta) = \mathbf{0}.$$

By the implicit function theorem [19], if

(a) (2.12) satisfies strong duality (*e.g.*, if Slater's condition holds);
(b) $g(\lambda(\theta), \mu(\theta), \theta) = \mathbf{0}$;
(c) g is continuously differentiable in (λ, μ, θ) in a neighborhood around $(\lambda(\theta), \mu(\theta), \theta)$; and
(d) $\frac{\partial g}{\partial(\lambda,\mu)}(\lambda(\theta), \mu(\theta), \theta)$ is invertible,

then the derivative $\frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta)$ exists and is given by

$$\begin{bmatrix} \frac{\mathrm{d}\lambda}{\mathrm{d}\theta}(\theta) \\ \frac{\mathrm{d}\mu}{\mathrm{d}\theta}(\theta) \end{bmatrix} = -\left[\frac{\partial g}{\partial(\lambda, \mu)}(\lambda(\theta), \mu(\theta), \theta)\right]^{-1} \frac{\partial g}{\partial \theta}(\lambda(\theta), \mu(\theta), \theta).$$

Assumption 7 directly satisfies condition (a), and the KKT equality conditions satisfy condition (b). Condition (c) is satisfied if $\tilde{h}_t$, $\frac{\partial \tilde{\ell}}{\partial \lambda}$, and $\frac{\partial \tilde{h}_t}{\partial \lambda}$ are continuously differentiable in (θ, λ) in a neighborhood around $(\theta, \lambda(\theta))$. Assumption 7.1 guarantees that this holds for $\frac{\partial \tilde{\ell}}{\partial \lambda}$, and Assumptions 7.2/7.3 guarantee that this holds for $\tilde{h}_t$ and $\frac{\partial h_t}{\partial \lambda}$.

Finally, note that

$$\frac{\partial g}{\partial(\lambda,\mu)}(\lambda(\theta),\mu(\theta),\theta) = \begin{bmatrix} \frac{\partial^2}{\partial\lambda^2}\tilde{\ell}(\theta,\lambda(\theta)) + \mu(\theta)\cdot\frac{\partial^2}{\partial\lambda^2}\tilde{h}_t(\theta,\lambda(\theta)) & \frac{\partial\tilde{h}_t}{\partial\lambda}(\theta,\lambda(\theta)) \\ \mu(\theta)\cdot\frac{\partial\tilde{h}_t}{\partial\lambda}(\theta,\lambda(\theta)) & \tilde{h}_t(\theta,\lambda(\theta)) - \alpha \end{bmatrix},$$

which Assumption 7.4 specifies is invertible, thus satisfying condition (d). □

We now briefly remark on Theorem 3(i)'s use of Assumption 6.4 and its relevance in practice. Recall that Assumption 6.4 assumes $g_{i,j} : \Theta \to \mathbb{R}$ is differentiable a.e. and its level sets have measure zero. The following two lemmas show that both of these conditions are easily satisfied if $g_{i,j}$ is a neural network with Lipschitz continuous activation functions and a bias term in its final output layer. For instance, in Example 2, $g_{i,j}(\theta) = f_\theta(X_i)_j$ is the output of a neural network f_θ (albeit a neural network with a more complex U-Net [223, 82] architecture than a MLP).

Lemma 6. *A multi-layer perceptron (MLP) with Lipschitz continuous activation functions is locally Lipschitz continuous and therefore differentiable a.e. with respect to its parameters.*

Specifically, let $g_K : \Theta \to \mathbb{R}^n$ denote a K-layer MLP for some $K \in \mathbb{N}$, defined recursively as

$$g_k(\theta) := g_k(\theta_{1:k}) := \sigma_k(W_k \cdot g_{k-1}(\theta_{1:k-1}) + b_k) \qquad \forall k \in [K]$$
$$g_0 \in \mathbb{R}^d \text{ is the fixed input to the MLP.}$$

Here, (W_k, b_k) are the weight matrix and bias vector in each layer k, and we use the shorthand notation $\theta_{1:k} := (W_{1:k}, b_{1:k})$, and $\theta := \theta_{1:K}$. If each σ_k is a Lipschitz continuous activation function, then g_K is locally Lipschitz continuous and therefore differentiable a.e. in θ.

Proof. We use induction on k. For the base case $k = 0$, g_0 is constant and therefore (locally) Lipschitz continuous.

Now, fix some θ and suppose g_{k-1} is locally Lipschitz continuous in a neighborhood $\mathcal{N}(\theta)$ around θ with local Lipschitz constant L_{k-1}:

$$\forall \theta' \in \mathcal{N}(\theta): \quad \|g_{k-1}(\theta') - g_{k-1}(\theta)\| \le L_{k-1}\|\theta' - \theta\|.$$

Define $\bar{n} := \sup_{\theta' \in \mathcal{N}(\theta)} \|\theta'\| < \infty$ and $\bar{g}_{k-1} := \|g_{k-1}(\theta)\| + L_{k-1} \operatorname{diam}(\mathcal{N}(\theta))$ so that $\|g_{k-1}(\theta')\| \leq \bar{g}_{k-1}$ for all $\theta' \in \mathcal{N}(\theta)$. Let M_k be the Lipschitz constant of σ_k. Then, for all θ, θ':

$$\begin{aligned}
&\|g_k(\theta') - g_k(\theta)\| \\
&= \left\|\sigma_k(W_k' \cdot g_{k-1}(\theta') + b_k') - \sigma_k(W_k \cdot g_{k-1}(\theta) + b_k)\right\| \\
&\leq M_k \left\|W_k' \cdot g_{k-1}(\theta') + b_k' - (W_k \cdot g_{k-1}(\theta) + b_k)\right\| \\
&\leq M_k \left(\left\|W_k' \cdot g_{k-1}(\theta') - W_k \cdot g_{k-1}(\theta)\right\| + \left\|b_k' - b_k\right\|\right) \\
&\leq M_k \left(\left\|W_k' \cdot g_{k-1}(\theta') - W_k \cdot g_{k-1}(\theta)\right\| + \|\theta' - \theta\|\right).
\end{aligned}$$

We have

$$\begin{aligned}
&\left\|W_k' \cdot g_{k-1}(\theta') - W_k \cdot g_{k-1}(\theta)\right\| \\
&= \left\|W_k' \cdot g_{k-1}(\theta') - W_k' \cdot g_{k-1}(\theta) + W_k' \cdot g_{k-1}(\theta) - W_k \cdot g_{k-1}(\theta)\right\| \\
&\leq \left\|W_k' \cdot g_{k-1}(\theta') - W_k' \cdot g_{k-1}(\theta)\right\| + \left\|W_k' \cdot g_{k-1}(\theta) - W_k \cdot g_{k-1}(\theta)\right\| \\
&\leq \left\|W_k'\right\| \|g_{k-1}(\theta') - g_{k-1}(\theta)\| + \left\|W_k' - W_k\right\| \|g_{k-1}(\theta)\| \\
&\leq \bar{n} \cdot L_{k-1} \|\theta' - \theta\| + \|\theta' - \theta\| \, \bar{g}_{k-1}.
\end{aligned}$$

Putting these results together yields

$$\|g_k(\theta') - g_k(\theta)\| \leq M_k(\bar{n} L_{k-1} + \bar{g}_{k-1} + 1) \|\theta' - \theta\|.$$

Thus, g_k is locally Lipschitz in θ on the neighborhood $\mathcal{N}(\theta)$ with Lipschitz constant $L_k = M_k(\bar{n} L_{k-1} + \bar{g}_{k-1} + 1)$.

Hence, by induction, g_K is locally Lipschitz in θ. Finally, by Rademacher's Theorem [33, Corollary 4.12], g_K is differentiable a.e. □

Lemma 7. *Let $W \subseteq \mathbb{R}^n$ be an open set, and let $h : W \to \mathbb{R}$ be a continuous function. If $g : W \times \mathbb{R} \to \mathbb{R}$ is defined by $g(w, b) := h(w) + b$, then for every $c \in \mathbb{R}$, the c-level set of g, $L_c := \{(w, b) \in W \times \mathbb{R} \mid g(w, b) = c\}$, has measure zero.*

Proof. First, we establish measurability of L_c. g is measurable because it is continuous, and $L_c = g^{-1}(\{c\})$ is the preimage of the Borel measurable singleton set $\{c\}$. Therefore, L_c is a measurable set, and its indicator function $\mathbf{1}_{L_c}$ is a measurable function.

For any $m \in \mathbb{N}$, let λ_m denote the Lebesgue measure on $\mathbb{R}^m$. Then,

$$\begin{aligned}
\lambda_{n+1}(L_c) &= \int_{\mathbb{R}^{n+1}} \mathbf{1}_{L_c}(w,b)\,\mathrm{d}\lambda_{n+1}(w,b) \\
&= \int_{\mathbb{R}^n}\int_{\mathbb{R}} \mathbf{1}_{L_c}(w,b)\,\mathrm{d}\lambda_1(b)\,\mathrm{d}\lambda_n(w) && \text{Tonelli's theorem} \\
&= \int_{\mathbb{R}^n} \lambda_1(\{b \in \mathbb{R} \mid g(w,b) = c\})\,\mathrm{d}\lambda_n \\
&= \int_{\mathbb{R}^n} \lambda_1(\{c - h(w)\})\,\mathrm{d}\lambda_n && \{b \mid g(w,b) = c\} = \{c - h(w)\} \\
&= \int_{\mathbb{R}^n} 0\,\mathrm{d}\lambda_n = 0 && \text{singleton sets have 0 measure}
\end{aligned}$$

□

Convexity of $\tilde{h}$

In this section, for ease of exposition, we write $\tilde{h}(\lambda, t)$ instead of $\tilde{h}_t(\lambda)$. The following proposition is used to show that (2.13) is convex.

Proposition 3. *Under the assumptions of Theorem 1 or Theorem 2, $\tilde{h}(\lambda, t)$ is nondecreasing in λ and convex in t. Furthermore, if $\{L_i\}_{i=1}^N$ and B are convex, then $\tilde{h}$ is jointly convex in (λ, t).*

Proof. First, we show that $\tilde{h}(\lambda, t)$ is nondecreasing in λ.

- In the setting of Theorem 1 (i.e., Assumptions 1 and 2), the functions $\{B, L_1, \ldots, L_N\}$ are all nondecreasing. Since ϕ is also nondecreasing, $\tilde{L}_{i,t}$ and $\tilde{B}_t$ are also nondecreasing. Thus, $\tilde{h}$ is nondecreasing in λ.
- In the setting of Theorem 2 (i.e., Assumptions 1 and 5 with CVaR risk), we showed in Section 2.8 that $\tilde{L}_{i,t}$ and $\tilde{B}_t$ are nondecreasing. Thus, $\tilde{h}$ is nondecreasing in λ.

Second, we show that $\tilde{h}(\lambda, t)$ is convex in t. We start by showing that $\tilde{B}_t(\lambda)$ is convex in t. For any $a \in [0, 1]$,

$$\begin{aligned}
\tilde{B}_{at+(1-a)t}(\lambda) &= at + (1-a)t + \phi(B(\lambda) - at + (1-a)t) \\
&= at + (1-a)t + \phi(a(B(\lambda) - t) + (1-a)(B(\lambda) - t)) \\
&\le at + (1-a)t + a\phi(B(\lambda) - t) + (1-a)\phi(B(\lambda) - t) && \phi \text{ is convex} \\
&= a\tilde{B}_t(\lambda) + (1-a)\tilde{B}_t(\lambda).
\end{aligned}$$

An identical argument shows that $\tilde{L}_{i,t}$ is convex in t. Since $\tilde{h}$ is the sum of functions that are convex in t, $\tilde{h}$ is convex in t.

If $\{L_i\}_{i=1}^N$ and B are convex (in λ), then by Lemma 3, $\{\tilde{L}_{i,t}\}_{i=1}^N$ and $\tilde{B}_t$ are all convex in (λ, t). Since $\tilde{h}$ is their mean, it is also convex in (λ, t). □

2.9 Experimental Details

Computational resources Our experiments were performed on two computers with the following hardware:

1. 2× AMD EPYC 7513 32-Core CPUs, 4× NVIDIA A100 GPUs (80GiB GPU memory each), 1 TiB RAM
2. Intel Core i9-12900KS CPU, 2× NVIDIA RTX A6000 GPUs (48GiB GPU memory each), 125 GiB RAM

Our code is written in Python and primarily relies on the PyTorch [202] deep learning library. We use the Adam optimizer [133].

Tumor image segmentation

Datasets We used images from 4 public datasets (CVC-ClinicDB [26], CVC-ColonDB [25], ETIS-LaribPolypDB [244], Kvasir-SEG [120]), yielding a total of 2,188 images with per-pixel binary mask labels. Following a similar setup to [82], we used 1,450 images from the CVC-ClinicDB and Kvasir-SEG datasets to form a training set; we used the same training split as [82]. From the remaining 738 images, we created 10 different val (400 images) / test (338 images) splits from different random seeds.

Note that unlike [82], we did include the "CVC-300" dataset in our test set, as "CVC-300" is a duplicate of CVC-ColonDB.

Model fine-tuning We use a pretrained PraNet [82] model, which features a Res2Net [93] backbone and a U-Net-like [223] decoder. During model fine-tuning, we only update the weights of the decoder; we keep the Res2Net backbone weights frozen. We tune the learning rate with a grid search over (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}). Models are trained with a batch size of 400 for up to 100 epochs with early-stopping after 10 epochs of no improvement on the val set.

We compare two different forms of fine-tuning, as shown in Figure 2.1.

- "cross-entropy" fine-tuning refers to using the same multi-scale binary cross-entropy loss that [82] used to pretrain the PraNet model.

- "conformal risk training" refers to using the gradient (2.9). Recall that Example 2 derives the exact expression $\frac{\mathrm{d}\lambda(\theta)}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta} f_\theta(X_i)_j$ where j is the index of some pixel in some image i of the training minibatch. This gradient may have high variance because it is the gradient through the model's output on exactly a single pixel among an entire minibatch of images. In practice, we reduce the variance of the gradient estimator by averaging the model's gradient over other pixels with similar model outputs:

$$\frac{\mathrm{d}\lambda(\theta)}{\mathrm{d}\theta} \approx \frac{1}{M} \sum_{(i,j)\in I} \frac{\mathrm{d}}{\mathrm{d}\theta} f_\theta(X_i)_j$$

  for the M indices $I = \{(i_k, j_k)\}_{k=1}^{M}$ whose values $f_\theta(X_{i_k})_{j_k}$ are closest to $\lambda(\theta)$. (We set M dynamically to be 0.5% of the number of positive pixels in each minibatch.) This approach is inspired by the variance reduction techniques introduced by [114, 190], which show reduced gradient estimation variance at the expense of possibly incurring some estimation bias. We leave it to future work to formally analyze bias-variance trade-off of this gradient estimator.

Computational efficiency Our conformal risk training procedure imposes negligible computational overhead compared to fine-tuning using the standard binary cross-entropy loss. For each minibatch of M images with d pixels each, the only overhead is incurred by the binary search procedure from Algorithm 2. In practice, we implement this by sorting the model outputs $\{f_\theta(X_i)_j\}_{i\in[M],\, j\in[d]}$, which requires $O(Md \log Md)$ time complexity and took on average <50 milliseconds of wall-clock time per minibatch.

Battery storage operations

Dataset We use the same dataset as Donti et al. [73] in our battery storage problem. In this dataset, the target $y \in \mathbb{R}^{24}$ is the hourly PJM day-ahead system energy price for 2011-2016, for a total of 2189 days. However, whereas Donti et al. [73] excluded any days whose electricity prices are too high (>$500/MWh), our conformal risk training method is specifically designed to handle tail-risk, so we did not exclude these "outliers." For predicting target for a given day, the inputs $x \in \mathbb{R}^{77}$ include the previous day's log-prices, the given day's hourly load forecast, the previous day's hourly temperature, and several calendar-based features such as whether the given day is a weekend or a US holiday.

Unlike Donti et al. [73], we did not include the given day's hourly temperature as input features, as such features are unavailable in practice (although temperature

forecasts may be available). To make the problem instance more challenging, we also added i.i.d. $\mathcal{N}(0, \sqrt{20})$ noise to the price targets.

We take a random 20% subset of the dataset as the test set; because the test set is selected randomly, it is considered exchangeable with the rest of the dataset. For each of 10 seeds, we further use a 65%/35% random split of the remaining data for training and calibration.

Decision constraints We note that we use a different, but equivalent, parameterization of the battery state-of-charge constraints. Donti et al. [73] use the task loss

$$f(y,z) = \sum_{t=1}^{T} y_t (z^{\text{in}} - z^{\text{out}})_t + \epsilon^{\text{flex}} \left\| z^{\text{state}} - \frac{C}{2}\mathbf{1} \right\|^2 + \epsilon^{\text{ramp}} \left(\left\| z^{\text{in}} \right\|^2 + \left\| z^{\text{out}} \right\|^2 \right)$$

with constraints, for all $t = 1, \ldots, T$:

$$\begin{aligned} &z_0^{\text{state}} = C/2, \quad z_t^{\text{state}} = z_{t-1}^{\text{state}} - z_t^{\text{out}} + \gamma z_t^{\text{in}}, \\ &0 \le z^{\text{in}} \le c^{\text{in}}, \quad 0 \le z^{\text{out}} \le c^{\text{out}}, \quad 0 \le z_t^{\text{state}} \le C. \end{aligned}$$

Instead of using a z^{state} variable to represent state of charge, we equivalently express the task loss and constraints in terms of a variable $z^{\text{net}} \in \mathbb{R}^T$ representing the difference of the state of charge from the starting point. Letting $z = (z^{\text{in}}, z^{\text{out}}, z^{\text{net}})$ denote the decision vector, we have

$$f(y,z) = y^\top (z^{\text{in}} - z^{\text{out}}) + \epsilon^{\text{flex}} \left\| z^{\text{net}} \right\|^2 + \epsilon^{\text{ramp}} \left(\left\| z^{\text{in}} \right\|^2 + \left\| z^{\text{out}} \right\|^2 \right)$$

with the constraint set $\mathcal{Z}$ from (2.11). This reparameterization is necessary for the decision variable z to satisfy our "decision-scaling" requirement: if $z \in \mathcal{Z}$, then $\forall \lambda \in [0,1], \lambda z \in \mathcal{Z}$.

Model and fine-tuning Our price forecasting model $\hat{y}_\theta$ is a fully-connected multilayer perceptron (MLP) with 3 hidden layers, each with 256 units, LeakyReLU activation, and batch normalization [115] layers. We use a batch size of 400.

We pretrain the MLP to predict prices using a mean squared error (MSE) loss. During pretraining, we tune the learning rate across (10^{-4}, $10^{-3.5}$, 10^{-3}, $10^{-2.5}$, 10^{-2}, $10^{-1.5}$), and we use L2 weight decay strength of 10^{-4}. We pretrain for up to 500 epochs with early-stopping after 20 epochs of no improvement on the val set.

During fine-tuning, we tune the learning rate with a grid search over (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}) and we keep the 10^{-4} L2 weight decay. We fine-tune for up to 100

epochs with early-stopping after 10 epochs of no improvement on the val set. During fine-tuning, we use a 90%/10% weighted combination of the fine-tuning loss and the MSE loss.

We compare two different forms of fine-tuning, as shown in Figure 2.2.

- "task loss" refers to fine-tuning the model $\hat{y}_\theta$ using the decision-focused learning task loss from [73]. As explained in Section 2.5, we define the decision for an input x as
$$\hat{z}_\theta(x) := \operatorname*{arg\,min}_{z \in \mathcal{Z}} f(\hat{y}_\theta(x), z),$$
and the task loss incurred is $f(y, \hat{z}_\theta(x))$. Thus, "task loss" fine-tuning refers to using $f(y, \hat{z}_\theta(x))$ as the training loss function.

- "conformal risk training" refers to using the gradient (2.9). In this battery storage problem, we seek to control the CVaR tail risk of the financial term $L(\theta, \lambda) = \lambda(\hat{z}^{\text{in}}_\theta(X) - \hat{z}^{\text{out}}_\theta(X))^\top Y$ while minimizing the task loss $\ell(\theta, \lambda) = f(Y, \lambda \hat{z}_\theta(X))$. We seek to solve
$$\begin{aligned} \min_{\theta \in \Theta,\, \lambda \in \Lambda} \quad & \mathbb{E}_{(X,Y)\sim\mathcal{P}}[f(Y, \lambda \hat{z}_\theta(X))] \\ \text{s.t.} \quad & \text{CVaR}^{\delta}_{(X,Y)\sim\mathcal{P}}[\lambda(\hat{z}^{\text{in}}_\theta(X) - \hat{z}^{\text{out}}_\theta(X))^\top Y] \le \alpha. \end{aligned}$$
During each minibatch of training, we pick the t that allows for the largest λ. In other words, we pick $\lambda(\theta)$ by solving the convex optimization problem
$$\lambda(\theta) = \operatorname*{arg\,max}_{\lambda \in \Lambda} \max_{t \in \mathbb{R}} \lambda \quad \text{s.t. } \tilde{h}_t(\theta, \lambda) \le \alpha,\ B(\lambda_{\min}) \le t \le \alpha. \tag{2.13}$$
This problem is convex because $\tilde{h}_t$ is jointly convex in (λ, t) by Proposition 3. The gradient term $\frac{\mathrm{d}\lambda(\theta)}{\mathrm{d}\theta}$ is then given by Theorem 3(ii) with the $\phi_{\text{CVaR}^\delta}$ disutility function.

Computational efficiency Our conformal risk training procedure imposes negligible computational overhead compared to fine-tuning using the task loss. For each minibatch, the main overhead incurred is from solving and differentiating through (2.13), which we found in practice to require on average <50 milliseconds of wall-clock time per minibatch using `cvxpylayers` [3].

2.10 Conformal Risk Training Subsumes Conformal Training

In this section, we illustrate how our method, conformal risk training, includes conformal training (ConfTr) from Stutz et al. [254] as a special case.

Consider the multi-class classification problem with classes $\mathcal{Y} = \{1, \dots, K\}$. For a nonconformity score function $s_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ parameterized by θ, conformal prediction [278, 233, 9] forms a prediction set

$$C_\theta(X; \lambda) = \{k \in \mathcal{Y} \mid s_\theta(X, k) \leq 1 - \lambda\}.$$

The overall goal posed by Stutz et al. [254] is to optimize inefficiency (*i.e.*, minimize the average size of the conformal prediction set) subject to a minimum coverage rate. The minimum coverage constraint can be expressed using the loss function $L(\theta, \lambda) := \mathbf{1}[s_\theta(X, Y) > 1 - \lambda]$, which is left-continuous and nondecreasing in λ. Since $\mathbb{E}[L(\theta, \lambda)] = \Pr(s_\theta(X, Y) > 1 - \lambda) = \Pr(Y \notin C_\theta(X; \lambda))$, the constraint $\mathbb{E}[L(\theta, \lambda)] \leq \alpha$ is equivalent to $\Pr(Y \in C_\theta(X; \lambda)) \geq 1 - \alpha$.

To optimize inefficiency, Stutz et al. [254] proposes using a cost function that is the sum of a soft assignment of classes:

$$\ell(\theta, \lambda) := \max\left(0, \sum_{k=1}^{K} \sigma\left(\frac{(1 - \lambda) - s_\theta(X, k)}{T}\right) - 1\right),$$

where T is a temperature hyperparameter. Thus, this problem is an instance of the end-to-end optimal risk control problem

$$\min_{\theta \in \Theta, \lambda \in \Lambda} \mathbb{E}[\ell(\theta, \lambda)] \quad \text{s.t. } \mathbb{E}[L(\theta, \lambda)] \leq \alpha.$$

Given a dataset $D = \{(X_i, Y_i)\}_{i=1}^N$ drawn exchangeably from $\mathcal{P}$, let $L_i(\theta, \lambda) := \mathbf{1}[s_\theta(X_i, Y_i) > 1 - \lambda]$. Clearly, every L_i is bounded by $B = 1$. Following the conformal risk control procedure, and noting that larger values of λ yield lower

objective values, we may pick

$$\begin{aligned}
\lambda(\theta) &:= \sup\left\{\lambda \in \mathbb{R} \;\middle|\; \frac{1}{N+1}\left(1 + \sum_{i=1}^{N} \mathbf{1}[s_\theta(X_i, Y_i) > 1 - \lambda]\right) \leq \alpha\right\} \\
&= \sup\left\{\lambda \in \mathbb{R} \;\middle|\; \frac{1}{N+1}\left(1 + \sum_{i=1}^{N} (1 - \mathbf{1}[s_\theta(X_i, Y_i) \leq 1 - \lambda])\right) \leq \alpha\right\} \\
&= \sup\left\{\lambda \in \mathbb{R} \;\middle|\; 1 - \frac{1}{N+1}\sum_{i=1}^{N} \mathbf{1}[s_\theta(X_i, Y_i) \leq 1 - \lambda] \leq \alpha\right\} \\
&= \sup\left\{\lambda \in \mathbb{R} \;\middle|\; \sum_{i=1}^{N} \mathbf{1}[s_\theta(X_i, Y_i) \leq 1 - \lambda] \geq (N+1)(1-\alpha)\right\} \\
&= 1 - s_{\theta,(\lceil (N+1)(1-\alpha) \rceil)}
\end{aligned} \tag{2.14}$$

where $s_{\theta,(j)}$ denotes the j-th smallest value of the set $S := \{s_\theta(X_i, Y_i)\}_{i=1}^{N} \cup \{\infty\}$. By Proposition 1, $\lambda(\theta)$ satisfies $\Pr(Y \in C_\theta(X; \lambda(\theta))) \geq 1 - \alpha$.

We note that whereas Stutz et al. [254] approximates $\frac{\mathrm{d}\lambda(\theta)}{\mathrm{d}\theta}$ using differentiable approximate ranking and sorting operations, the gradient can be computed exactly [114, 311, 190]. That is, if $s_\theta(X_i, Y_i)$ is differentiable w.r.t. θ for all $i = 1, \ldots, N$ and if $s_{\theta,(\lceil (N+1)(1-\alpha) \rceil)}$ is unique amongst the calibration scores, then by [311, Theorem 2]

$$\frac{\mathrm{d}\lambda(\theta)}{\mathrm{d}\theta} = \begin{cases} -\frac{\mathrm{d}}{\mathrm{d}\theta} s_\theta(X_{\sigma(k)}, Y_{\sigma(k)}), & \text{if } \alpha \geq \frac{1}{N+1} \\ 0, & \text{otherwise} \end{cases}$$

where $k = \lceil (N+1)(1-\alpha) \rceil$ and $\sigma : [N] \to [N]$ denotes the permutation that sorts the scores in ascending order, such that $s_\theta(X_{\sigma(i)}, Y_{\sigma(i)}) \leq s_\theta(X_{\sigma(j)}, Y_{\sigma(j)})$ for all $i < j$.

Chapter 3

END-TO-END CONFORMAL CALIBRATION FOR OPTIMIZATION UNDER UNCERTAINTY

The previous chapter presented a general framework for combining conformal risk control guarantees with end-to-end learning using differentiable optimization, and it demonstrated how the framework can be instantiated for prediction models that output a single point estimate. This present chapter focuses on another specific instance of the framework: learning set-valued uncertainty representations for conditional robust optimization. Here, the risk guarantee is a coverage guarantee on the uncertainty sets, achieved using conformal prediction, which is a special case of conformal risk control. In addition, we propose to represent general convex uncertainty sets with partially input-convex neural networks, which are learned as part of our framework. Our approach consistently improves upon two-stage estimate-then-optimize baselines on concrete applications in energy storage arbitrage and portfolio optimization.

This chapter is primarily based on the following paper:

[311] Christopher Yeh*, Nicolas Christianson*, Alan Wu, Adam Wierman, and Yisong Yue. 2025. End-to-end conformal calibration for optimization under uncertainty. *Transactions on Machine Learning Research*, (Dec. 2025). https://openreview.net/forum?id=yM8qkT0f9H.

3.1 Introduction

Well-calibrated estimates of forecast uncertainty are vital for risk-aware decision-making in many real-world systems. For instance, grid-scale battery operators forecast electricity prices to schedule battery charging/discharging to maximize profit, while accounting for forecast uncertainty to mitigate financial or operational risk. Similarly, financial investors use forecasts of asset returns with uncertainty estimates to maximize portfolio returns while minimizing downside risk.

Traditional approaches to decision-making under uncertainty often follow an "estimate-then-optimize" (ETO) paradigm [55], in which the uncertainty estimation and decision-making stages are decoupled. In the "estimation" stage, a predictive

[1] https://creativecommons.org/licenses/by/4.0/

model is trained to forecast a target quantity along with an uncertainty set. In the "optimization" stage, this uncertainty estimate informs a downstream decision. Crucially, the cost or performance of the downstream decision is typically not fed back into the training of the predictive model.

A recent line of work [55, 203, 281, 56] has made steps toward bridging the gap between uncertainty quantification and robust optimization-driven decision-making, where optimization problems take a forecast uncertainty set as a parameter, as is common in energy systems [302, 201] and financial applications [99, 213]. However, existing approaches are suboptimal for several reasons:

1. **Lack of decision-aware training:** The predictive model is typically not trained with feedback from the downstream objective. Because the downstream objective is often asymmetric with respect to the forecasting model's errors, prediction models trained with decision-agnostic losses may achieve high predictive accuracy but still perform poorly on the decision-making objective.
2. **Restricted uncertainty set parameterizations:** To preserve tractability of the robust optimization problem, uncertainty sets are often constrained to simple parametric forms (e.g., boxes or ellipsoids), limiting the expressivity of uncertainty estimates.
3. **Calibration challenges:** Because neural network models are often poor at estimating their own uncertainty, the forecasts may not be well-calibrated. Recent approaches such as isotonic regression [142] and conformal prediction [233] have made progress in providing *calibrated* uncertainty estimates from deep learning models, but such methods are typically applied post-hoc to trained models and are therefore difficult to incorporate into an end-to-end training procedure.

As such, there is as of yet *no* comprehensive methodology for training calibrated uncertainty-aware deep learning models end-to-end with downstream decision-making objectives. In this work, we provide the first such methodology. We make three specific contributions corresponding to the three issues identified above:

1. **We develop a framework for training prediction models end-to-end with downstream decision-making objectives and conformal-calibrated uncertainty sets in the context of the *conditional robust optimization* problem.** This framework is illustrated in Figure 3.1 (bottom). By including differentiable conformal calibration in our model during training, we close the loop and ensure that feedback from the uncertainty's impact on the downstream

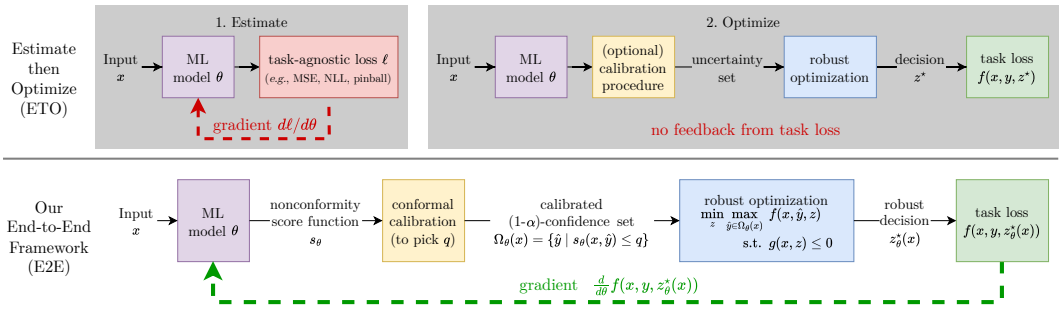


Figure 3.1: Whereas prior "estimate-then-optimize" (ETO, top) approaches separate the model training from the optimization (decision-making) procedure, we propose a framework for end-to-end (E2E, bottom) conformal calibration for optimization under uncertainty that directly trains the machine learning model using gradients from the task loss.

objective is accounted for in the training process, since not all model errors nor uncertainty estimates will result in the same downstream cost. This end-to-end training enables the model to focus its learning capacity on minimizing error and uncertainty on outputs with the largest decision-making cost, with more leeway for outputs that have lower costs.

2. **We propose using partially input-convex neural networks (PICNNs) as the nonconformity score function for conformal prediction, enabling the approximate parametrization of arbitrary compact, convex uncertainty sets in the conditional robust optimization problem.** To the best of our knowledge, no existing works use PICNNs to parametrize such arbitrary convex uncertainty sets. Due to the universal convex function approximation property these networks enjoy [54], this approach enables training much more general representations of uncertainty than prior works have considered, which in turn yields substantial improvements on downstream decision-making performance. Importantly, PICNNs are well-matched to our conditional robust optimization problem: we show that the robust problem resulting from this parametrization can be reformulated as a tractable convex optimization problem.
3. **We propose an exact and computationally efficient method to differentiate through the conformal prediction procedure during training.** Unlike prior work [254], our method gives exact gradients, without using approximate ranking and sorting methods.

Finally, we evaluate the performance of our approach on two applications: an energy storage arbitrage task and a portfolio optimization problem. We demonstrate that the combination of end-to-end training with the flexibility of the PICNN-based

uncertainty sets consistently improves over ETO baseline methods. The performance benefit of our end-to-end method is apparent even under distribution shift.

3.2 Problem Statement and Background

Our problem is defined formally as follows: suppose that data $(x, y) \in \mathbb{R}^m \times \mathbb{R}^n$ is sampled i.i.d. from an unknown joint distribution $\mathcal{P}$. Upon observing the input x (but not the label y), an agent makes a decision $z \in \mathbb{R}^p$. After the decision is made, the true label y is revealed, and the agent incurs a *task loss* $f(x, y, z)$, for some known task loss function $f : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$. In addition, the agent's decision must satisfy a set of joint constraints $g(x, z) \leq 0$ coupling x and z.

As an illustrative example, consider an agent who would like to minimize the costs of charging and discharging a battery over 24 hours in a day. The agent may use weather forecasts and historical observations x to predict future energy prices y. Based on the predicted prices, the agent decides on the amount z to charge or discharge the battery. The task loss f is the cost incurred by the agent, and the constraints g include limits on how fast the battery can charge as well as the maximum capacity of the battery. This example is explored in more detail in Section 3.5.

Because the agent does not observe the label y prior to making its decision, ensuring good performance and constraint satisfaction requires that the agent makes decisions z that are *robust* to the various outcomes of y. A common objective is to choose z to robustly minimize the task loss and satisfy the constraints over all realizations of y within a $(1 - \alpha)$-confidence region $\Omega(x) \subset \mathbb{R}^n$ of the true conditional distribution $\mathcal{P}(y \mid x)$, where $\alpha \in (0, 1)$ is a fixed risk level based on operational requirements. In this case, the agent's robust decision can be expressed as the optimal solution to the following **conditional robust optimization (CRO)** problem [55]:

$$z^\star(x) := \operatorname*{arg\,min}_{z \in \mathbb{R}^p} \max_{\hat{y} \in \Omega(x)} f(x, \hat{y}, z) \text{ s.t. } g(x, z) \leq 0. \tag{3.1}$$

After the agent decides $z^\star(x)$, the true label y is revealed, and the agent incurs the task loss $f(x, y, z^\star(x))$. Thus, the agent seeks to minimize expected task loss

$$\mathbb{E}_{(x,y)\sim\mathcal{P}} \left[f(x, y, z^\star(x)) \right]. \tag{3.2}$$

While the joint distribution $\mathcal{P}$ is unknown, we assume that we have a dataset $D = \{(x_i, y_i)\}_{i=1}^N$ of i.i.d. samples from $\mathcal{P}$. Then, our objective is to train a machine learning model to learn an approximate $(1 - \alpha)$-confidence set $\Omega(x)$ of possible y values for each input x. Formally, our learned $\Omega(x)$ should satisfy the following *marginal coverage* guarantee.

Definition 4 (marginal coverage)**.** *An uncertainty set* $\Omega(x)$ *for the distribution* $\mathcal{P}$ *provides* marginal coverage at level $(1-\alpha)$ *if* $\mathbb{P}_{(x,y)\sim\mathcal{P}}\left(y \in \Omega(x)\right) \geq 1-\alpha$.

Comparison to related work. The problem of constructing data-driven and machine-learned uncertainty sets with probabilistic coverage guarantees for use in robust optimization has been widely explored in prior literature (e.g., Bertsimas and Thiele [32], Bertsimas et al. [29], Alexeenko and Bitar [5], and Goerigk and Kurtz [97]). Chenreddy et al. [55] first coined the phrase "conditional robust optimization" for the problem (3.1) and considered learning context-dependent uncertainty sets $\Omega(x)$ in this setting. However, their approach results in a mixed integer optimization that is intractable to solve for large-scale problems. Moreover, they follow the "estimate then optimize" (ETO) paradigm [79]. As shown in Figure 3.1 (top), the ETO paradigm separates the machine learning model training from the decision optimization. The lack of feedback from the downstream task loss during model training in ETO generally leads to uncertainty sets $\Omega(x)$ which yield suboptimal results. Several other recent papers follow the ETO paradigm using homoskedastic ellipsoidal uncertainty sets [123], heteroskedastic box and ellipsoidal uncertainty sets [256], and a "union of balls" parametrization of uncertainty [203]. In our experiments (Section 3.5), we demonstrate consistent improvements over the methods of Johnstone and Cox [123] and Sun et al. [256].

Closest to our work is an "end-to-end" formulation of the CRO problem posed by Chenreddy and Delage [56], which aims to learn conditional uncertainty sets $\Omega(x)$ using a weighted combination of the downstream task loss along with a "conditional coverage loss" to promote calibrated uncertainty. However, they focus solely on ellipsoidal uncertainty sets, and their conditional coverage loss does not provably ensure coverage for their learned uncertainty sets. In our experiments, we do not compare against Chenreddy and Delage [56] because we only consider other methods with a provable coverage guarantee.[2]

A concurrent related work by Wang et al. [281] approaches the problem of learning *unconditional* uncertainty sets for robust optimization, while achieving robust constraint satisfaction guarantees. While their method also uses an end-to-end task loss, we find their use of the same uncertainty set Ω for every problem instance

[2] As of the time of writing, the approach of Chenreddy and Delage [56] also suffers from substantial inconsistencies between their code implementation and the equations from their paper. In particular, the conditional coverage loss proposed in their paper is not implementable, as it will (almost surely) yield zero gradients.

(i.e., Ω is independent of x) to be highly restrictive. For example, in the context of our battery control problem, this restriction would disallow the use of weather forecasts and historical price data to estimate uncertainty in future energy prices.[3] They also use restrictive uncertainty set parametrizations such as box, ellipsoidal, and polyhedral uncertainty. Moreover, their guarantee on constraint satisfaction requires an assumption on the convergence of their algorithm to an approximate stationary point; however, this convergence is only guaranteed asymptotically, and not in finitely many samples.

An alternative line of research designs post-hoc conformal prediction sets aimed at minimizing decision costs in robust classification problems [135, 60]. However, to date, it is unclear how to adapt these methods, which produce discrete uncertainty sets, for regression settings.

In contrast, our work overcomes these limitations: we incorporate differentiable conformal calibration *during training* to ensure that uncertainty is learned end-to-end in a manner that is both calibrated and minimizes task loss. We apply split conformal post-hoc calibration during inference for provable guarantees on coverage. Furthermore, we use partially input-convex neural networks [8] to directly parameterize the nonconformity score function in conformal prediction, enabling a general and expressive representation of arbitrary conditional convex uncertainty regions that can vary with x and be used efficiently in robust optimization.

Beyond the above closely related work, this chapter builds upon and contributes to several different areas in machine learning and robust optimization; see Section 3.8 for a comprehensive discussion.

3.3 End-to-End Training of Conformally Calibrated Uncertainty Sets

In this section, we describe our proposed methodological framework for end-to-end task-aware training of predictive models with conformally calibrated uncertainty for the conditional robust optimization problem (3.1). Our overarching goal is to learn uncertainty sets $\Omega(x)$ which provide $(1-\alpha)$ coverage for any choice of $\alpha \in (0, 1)$, and which offer the lowest possible task loss (3.2). To this end, we must consider three primary questions:

1. How should the family of uncertainty sets $\Omega(x)$ be parametrized?

[3] After the present paper's submission, the authors of Wang et al. [281] updated their preprint and expanded their formulation to accommodate conditional uncertainty sets.

2. How can we guarantee that the uncertainty set $\Omega(x)$ provides coverage at level $1-\alpha$?
3. How can the uncertainty set $\Omega(x)$ be learned to minimize expected task loss?

Figure 3.1 (bottom) illustrates the key parts of our framework to answer these questions. First, we use a machine learning model to parametrize a nonconformity score function s_θ, and we define the uncertainty set $\Omega(x)$ to be a q-sublevel set of $s_\theta(x,\cdot)$. Second, we use conformal calibration to pick q to enforce marginal coverage. Third, we backpropagate gradients through both the robust optimization and conformal calibration steps to update the machine learning model, thereby enabling end-to-end learning. Section 3.3 describe each of these parts in detail, and Algorithm 4 shows pseudocode for both training and inference.

For the rest of the paper, we make the following assumptions on the functions f and g to ensure tractability of the resulting optimization problem.

Assumption 8. *We assume the task loss has the form $f(x,y,z) = y^\top F(x,z) + \tilde{f}(x,z)$, where $F(x,z)$ is an affine function of z and $\tilde{f}(x,z)$ is convex in z. Furthermore, we assume that $g(x,z)$ is convex in z.*

Technically, the assumption on f can be relaxed to the more general setting where f is a *conic-representable saddle function* that is convex in z and concave in y [124], which admits an automated dual representation [230]. However, we believe the core ideas for our exposition and proofs are much more straightforward with the stronger assumption in Assumption 8.

Representations of the uncertainty set

We consider convex uncertainty sets of the form

$$\Omega_\theta(x) = \{\hat{y} \in \mathbb{R}^n \mid s_\theta(x,\hat{y}) \leq q\}, \tag{3.3}$$

where $s_\theta : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ is an arbitrary *nonconformity score function* that is convex in $\hat{y}$, q is a scalar, and θ collects the parameters of a model that we will seek to learn. Note that this representation loses no generality; *any* family of convex sets $\Omega(x)$ can be represented as such a collection of sublevel sets of a partially input-convex function $s(x,\hat{y})$. This particular representation is chosen due to the ease of calibrating sets of this form via conformal prediction to ensure marginal coverage, as we will describe in Section 3.3.

In choosing a particular score function s_θ, one must balance two considerations: first, the *generality* of the sets $\Omega_\theta(x)$ that s_θ can represent, and second, the *tractability* of the resulting robust optimization problem (3.1). We will now show that our representation (3.3) generalizes commonly-used box and ellipsoidal uncertainty sets, which are known to have tractable robust problems; later, in Section 3.4, we will propose to approximate more general convex uncertainty sets using partially input-convex neural networks.

Box uncertainty sets. A simple uncertainty representation is box uncertainty where $\Omega(x) = [\underline{y}(x), \overline{y}(x)]$ is an n-dimensional box whose lower and upper bounds depend on x. Let $h_\theta : \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^n$ be a neural network that estimates lower and upper bounds: $h_\theta(x) = (h_\theta^{\text{lo}}(x), h_\theta^{\text{hi}}(x))$. To represent a box uncertainty set in the form (3.3), we define a nonconformity score function that generalizes scalar conformalized quantile regression [222]:

$$s_\theta(x, y) = \max(\left\|h_\theta^{\text{lo}}(x) - y\right\|_\infty, \left\|y - h_\theta^{\text{hi}}(x)\right\|_\infty).$$

Then, the uncertainty set (3.3) becomes

$$\Omega_\theta(x) = \left[h_\theta^{\text{lo}}(x) - q\mathbf{1},\ h_\theta^{\text{hi}}(x) + q\mathbf{1}\right] =: \left[\underline{y}(x),\ \overline{y}(x)\right].$$

Given a box uncertainty set $\Omega_\theta(x)$, we can take the dual of the inner maximization problem (see Section 3.9) to transform the robust optimization problem (3.1) into an equivalent form that is convex, and hence tractable, under Assumption 8:

$$\begin{aligned} z_\theta^\star(x) = \operatorname*{arg\,min}_{z \in \mathbb{R}^p} \min_{\nu \in \mathbb{R}^n} \quad & (\overline{y}(x) - \underline{y}(x))^\top \nu + \underline{y}(x)^\top F(x, z) + \tilde{f}(x, z) \\ \text{s.t.} \quad & \nu \geq \mathbf{0},\ \nu - F(x, z) \geq \mathbf{0},\ g(x, z) \leq 0. \end{aligned} \tag{3.4}$$

Ellipsoidal uncertainty sets. Another common form of uncertainty set is ellipsoidal uncertainty. Suppose a neural network model $h_\theta : \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{S}_+^n$ predicts mean and covariance parameters $h_\theta(x) = (\mu_\theta(x), \Sigma_\theta(x))$, so that $\hat{\mathcal{P}}(y \mid x; \theta) = \mathcal{N}(y \mid \mu_\theta(x), \Sigma_\theta(x))$ denotes a predicted conditional density, where $\mathcal{N}(\cdot \mid \mu, \Sigma)$ is the multivariate normal density function. In this case, we define the nonconformity score function based on the squared Mahalanobis distance [123, 256]

$$s_\theta(x, y) = (y - \mu_\theta(x))^\top (\Sigma_\theta(x))^{-1} (y - \mu_\theta(x)),$$

which yields uncertainty sets (3.3) that are ellipsoidal:

$$\Omega_\theta(x) = \{\hat{y} \mid (\hat{y} - \mu_\theta(x))^\top (\Sigma_\theta(x))^{-1} (\hat{y} - \mu_\theta(x)) \leq q\}.$$

Let $L_\theta(x)$ denote the unique lower-triangular Cholesky factor of $\Sigma_\theta(x)$ (i.e., $\Sigma_\theta(x) = L_\theta(x)L_\theta(x)^\top$). Taking the dual of the inner maximization problem and invoking strong duality (see Section 3.9), we transform the robust optimization problem (3.1) into an equivalent form that is convex, and hence tractable, under Assumption 8:

$$\begin{aligned} z_\theta^\star(x) = \underset{z\in\mathbb{R}^p}{\arg\min} \quad & \sqrt{q}\|L_\theta(x)^\top F(x,z)\|_2 + \mu_\theta(x)^\top F(x,z) + \tilde{f}(x,z) \\ \text{s.t.} \quad & g(x,z) \leq 0, \end{aligned} \tag{3.5}$$

Conformal uncertainty set calibration

As long as the uncertainty set $\Omega_\theta(x)$ can be expressed in the form (3.3), we can use the split conformal prediction procedure at inference time to choose a value q that ensures $\Omega_\theta(x)$ provides marginal coverage (Definition 4) at any confidence level $1-\alpha$. The split conformal procedure assumes access to a calibration dataset $D_{\text{cal}} = \{(x_i, y_i)\}_{i=1}^M$ drawn exchangeably from $\mathcal{P}$. We refer readers to Angelopoulos and Bates [9] for details on this procedure.

Lemma 8 (from Angelopoulos and Bates [9], Appendix D)**.** *Let $D_{cal} = \{(x_i, y_i)\}_{i=1}^M$ be a calibration dataset drawn exchangeably (e.g., i.i.d.) from $\mathcal{P}$, and let $s_i = s_\theta(x_i, y_i)$. If $q = \textsc{Quantile}(\{s_i\}_{i=1}^M,\ 1-\alpha)$ (see Algorithm 4) is the empirical $\frac{\lceil (M+1)(1-\alpha)\rceil}{M}$-quantile of the set $\{s_i\}_{i=1}^M$ and (x, y) is drawn exchangeably with D_{cal}, then $\Omega_\theta(x)$ has the marginal coverage guarantee*

$$1-\alpha \leq \mathbb{P}_{x,y,D_{cal}}(y \in \Omega_\theta(x)) \leq 1-\alpha+\frac{1}{M+1}.$$

We use split conformal prediction, rather than full conformal prediction, both for computational tractability and to avoid the problem of nonconvex uncertainty sets that can arise from the full conformal approach, as noted in Johnstone and Cox [123]. For the rest of this chapter, we assume $\alpha \in [\frac{1}{M+1}, 1)$ so that $q = \textsc{Quantile}(\{s_i\}_{i=1}^M,\ 1-\alpha) < \infty$ is finite. Thus, for appropriate choices of the score function s_θ, the uncertainty set $\Omega_\theta(x)$ is not unbounded.

While the split conformal prediction procedure in Lemma 8 ensures that the uncertainty set $\Omega_\theta(x)$ satisfies $(1-\alpha)$ coverage at inference time, this process does not address the question of *training* the uncertainty set $\Omega_\theta(x)$ (via the score function s_θ) to ensure optimal task performance while maintaining coverage. In Section 3.3, we propose applying a separate differentiable conformal prediction procedure during training to address this challenge.

Algorithm 4 End-to-end conformal calibration for robust decisions under uncertainty

function TRAIN(training data $D = \{(x_i, y_i)\}_{i=1}^N$, uncertainty level α, initial model parameters θ)
  for mini-batch $B \subset \{1, \ldots, N\}$ **do**
    Randomly split batch: $B = (B_{\text{cal}}, B_{\text{pred}})$
    Compute $q = \text{QUANTILE}(\{s_\theta(x_i, y_i)\}_{i \in B_{\text{cal}}},\ 1 - \alpha)$
    for $i \in B_{\text{pred}}$ **do**
      Solve for robust decision $z_\theta^\star(x_i)$ using (3.4), (3.5), or (3.8)
      Compute gradient of task loss: $d\theta_i = \partial f(x_i, y_i, z_\theta^\star(x_i))/\partial\theta$
    Update θ using gradients $\sum_{i \in B_{\text{pred}}} d\theta_i$

function INFERENCE(model parameters θ, calibration data $D_{\text{cal}} = \{(x_i, y_i)\}_{i=1}^M$, uncertainty level α, input x)
  Compute $q = \text{QUANTILE}(\{s_\theta(\tilde{x}, \tilde{y})\}_{(\tilde{x}, \tilde{y}) \in D_{\text{cal}}},\ 1 - \alpha)$
  return robust decision $z_\theta^\star(x)$ using (3.4), (3.5), or (3.8)

function QUANTILE(scores $S = \{s_i\}_{i=1}^M$, level β)
  $s_{(1)}, \ldots, s_{(M+1)} = \text{SORTASCENDING}(S \cup \{+\infty\})$
  return $s_{(\lceil (M+1)\beta \rceil)}$

End-to-end training and calibration

Thus far, we have discussed how to calibrate an uncertainty set $\Omega_\theta(x)$ of the form (3.3) to ensure coverage, and we described two choices of score function s_θ parametrizing common box and ellipsoidal uncertainty sets. However, to ensure that the uncertainty sets $\Omega_\theta(x)$ *both* guarantee coverage *and* ensure optimal downstream task performance, it is necessary to design an end-to-end training methodology that can incorporate both desiderata in a fully differentiable manner. We propose such a methodology in Algorithm 4.

Our end-to-end training approach minimizes the empirical task loss

$$\ell(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell_i(\theta)$$

using minibatch gradient descent, where $\ell_i(\theta) = f(x_i, y_i, z_\theta^\star(x_i))$. This requires differentiating through both the robust optimization problem as well as the conformal prediction step. The gradient of the task loss on a single instance is $\frac{\mathrm{d}\ell_i}{\mathrm{d}\theta} = \frac{\partial f}{\partial z}(x_i, y_i, z_\theta^\star(x_i)) \cdot \frac{\partial z_\theta^\star}{\partial \theta}(x_i)$, where $\frac{\partial z_\theta^\star}{\partial \theta}(x_i)$ is computed by differentiating through the Karush–Kuhn–Tucker (KKT) conditions of the convex reformulation of the optimization problem (3.1) (i.e., the problems (3.4), (3.5)) following the approach of Amos and Kolter [7], under mild assumptions on the differentiability of f and g.

Note that the gradient of any convex optimization problem can be computed with respect to its parameters as such [3, Appendix B].

To include calibration during training, we assume that for every (x, y) in our training set, $s_\theta(x, y)$ is differentiable w.r.t. θ almost everywhere; this assumption holds for common nonconformity score functions, including those used in this chapter. We then adopt the conformal training approach [254] in which a separate q is chosen in each minibatch, as shown in Algorithm 4. The chosen q depends on θ (through s_θ), and $z^\star_\theta(x_i)$ depends on the chosen q. Therefore $\frac{\partial z^\star_\theta}{\partial \theta}$ involves calculating $\frac{\partial z^\star_\theta}{\partial q}\frac{\partial q}{\partial \theta}$, where $\frac{\partial q}{\partial \theta}$ requires differentiating through the empirical quantile function. Whereas Stutz et al. [254] uses a smoothed approximate quantile function for calculating q, we find the smoothing unnecessary, as the gradient of the empirical quantile function is unique and well-defined almost everywhere. Importantly, our exact gradient is both more computationally efficient and simpler to implement than the smoothed approximate quantile approach. See Section 3.10 for more details.

After training has concluded and we have performed the final conformal calibration step, the resulting model enjoys the following theoretical guarantee on performance (*cf.* Sun et al. [256, Proposition 1]).

Proposition 4. *Under the same assumptions as Lemma 8, the task loss satisfies the following bound with probability at least* $1 - \alpha$ *(over* x, y*, and the calibration set* D_{cal}*):*

$$f(x, y, z^\star_\theta(x)) \leq \left(\min_{z \in \mathbb{R}^p} \max_{\hat{y} \in \Omega_\theta(x)} f(x, \hat{y}, z) \text{ s.t. } g(x, z) \leq 0 \right). \tag{3.6}$$

Proof. When $y \in \Omega_\theta(x)$, the definition of $z^\star_\theta(x)$ (3.1) guarantees that (3.6) holds. By Lemma 8, $\mathbb{P}_{x,y,D_{\text{cal}}}(y \in \Omega_\theta(x)) \geq 1 - \alpha$. □

We note that in practice, training models from scratch using only the task loss sometimes leads to poor local optima. Instead, in our experiments, we pretrain our models with a standard loss function (e.g., negative log-likelihood for the mean-variance predictor used in ellipsoidal uncertainty sets), and then we fine-tune the model using the task loss. This is a standard heuristic commonly used in the decision-focused learning and predict-then-optimize literature [73, 175]. Section 3.11 provides more details on our training procedure.

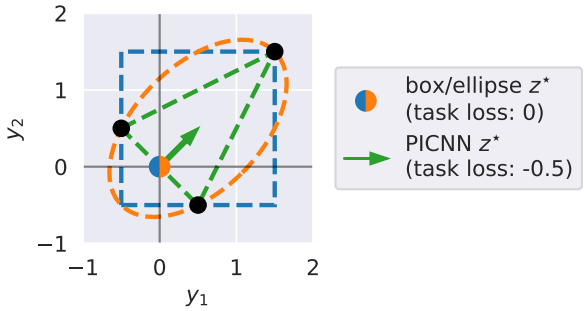


Figure 3.2: Consider a robust portfolio optimization problem with 2 assets, where $y \in \mathbb{R}^2$ is a random vector of asset returns, and the decision $z \in \mathbb{R}^2$ represents portfolio weights: $\max_z \min_{\hat{y}\in\Omega} -z^\top \hat{y}$ s.t. $z \geq 0$, $\mathbf{1}^\top z \leq 1$. Let the distribution of asset returns y be uniform over 3 discrete points (black). The optimal box (blue), ellipse (orange), and PICNN (green) uncertainty sets are shown with their robust decision vectors $z^\star$. The flexibility of the PICNN uncertainty representation allows it to achieve the lowest expected task loss.

3.4 Representing General Convex Uncertainty Sets via PICNNs

The previous section discussed how to train calibrated box and ellipsoidal uncertainty sets end-to-end to optimize the downstream task loss. However, both box and ellipsoidal uncertainty sets have restrictive shapes which may yield suboptimal task performance. If $\Omega_\theta(x)$ could represent *any* arbitrary convex uncertainty set, this more expressive class would enable obtaining better task loss. Figure 3.2 illustrates an example where a general convex uncertainty set representation provides a clear advantage over box and ellipsoid uncertainty.

To this end, we propose to directly learn a partially-convex nonconformity score function $s_\theta : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ that is convex only in the second input vector. Fixing x, any q-sublevel set $\{\hat{y} \in \mathbb{R}^n \mid s_\theta(x, \hat{y}) \leq q\}$ of s_θ is a convex set, and likewise every family of convex sets can be expressed as the q-sublevel sets of some partially-convex function. To implement this approach, we are faced with two questions.

1. How should we parametrize the score function s_θ so $\Omega_\theta(x)$ can approximate arbitrary *convex sets?* A natural answer is to parametrize s_θ with a partially input-convex neural network (PICNN) [8], which can efficiently approximate any partially-convex function [54]. We consider a PICNN defined as $s_\theta(x, y) = W_L\sigma_L + V_L y + b_L$, where

$$\begin{aligned} \sigma_0 &= \mathbf{0}, \qquad u_0 = x, & W_l &= \bar{W}_l \operatorname{diag}([\hat{W}_l u_l + w_l]_+) \\ u_{l+1} &= \mathrm{ReLU}\left(R_l u_l + r_l\right), & V_l &= \bar{V}_l \operatorname{diag}(\hat{V}_l u_l + v_l) \\ \sigma_{l+1} &= \mathrm{ReLU}\left(W_l\sigma_l + V_l y + b_l\right), & b_l &= \bar{B}_l u_l + \bar{b}_l, \end{aligned} \tag{3.7}$$

with weights $\theta = (R_l, r_l, \bar{W}_l, \hat{W}_l, w_l, \bar{V}_l, \hat{V}_l, v_l, \bar{B}_l, \bar{b}_l)_{l=0}^{L}$. The matrices $\bar{W}_l$ are constrained to be entrywise nonnegative to ensure convexity of s_θ with respect to y. For ease of notation, we assume all hidden layers $\sigma_1, \ldots, \sigma_L$ have the same dimension d.

2. *Does the chosen parametrization of $\Omega_\theta(x)$ (via PICNNs) yield a tractable reformulation of the CRO problem* (3.1)*?* Fortunately, we show in the following theorem that the answer is yes.

Theorem 4. *Let $\Omega_\theta(x) = \{\hat{y} \in \mathbb{R}^n \mid s_\theta(x, \hat{y}) \leq q\}$, where s_θ is a PICNN as defined in* (3.7)*. Then, under Assumption 8, the CRO problem* (3.1) *with uncertainty set $\Omega_\theta(x)$ is equivalent to the following convex (and hence tractable) minimization problem:*

$$\begin{aligned} z_\theta^\star(x) = \underset{z \in \mathbb{R}^p}{\arg\min} \; \min_{\nu \in \mathbb{R}^{2Ld+1}} \quad & b(\theta, q)^\top \nu + \tilde{f}(x, z) \\ \text{s.t.} \quad & A(\theta)^\top \nu = \begin{bmatrix} F(x, z) \\ \mathbf{0} \end{bmatrix}, \; \nu \geq \mathbf{0}, \; g(x, z) \leq 0 \end{aligned} \tag{3.8}$$

where $A(\theta) \in \mathbb{R}^{(2Ld+1)\times(n+Ld)}$ and $b(\theta, q) \in \mathbb{R}^{2Ld+1}$ are constructed from the weights θ of the PICNN (3.7)*, and b also depends on q.*

We prove Theorem 4 in Section 3.9; the main idea is that when $\Omega_\theta(x)$ is a sublevel set of a PICNN, we can equivalently reformulate the inner maximization problem in (3.1) as a linear program and take the dual to yield a tractable minimization problem.

Since the PICNN uncertainty sets are of the same form as (3.3) and yield a tractable convex reformulation (3.8) of the CRO problem (3.1), we can apply the split conformal procedure detailed in Section 3.3 to choose $q \in \mathbb{R}$ and obtain coverage guarantees on $\Omega_\theta(x)$, and we can employ the same end-to-end training methodology from Section 3.3 to train calibrated uncertainties end-to-end using the downstream task loss. In some cases during training, the inner maximization problem of (3.1) with PICNN-parametrized uncertainty set may be unbounded (if $\Omega_\theta(x)$ is not compact) or infeasible (if the chosen q is too small causing $\Omega_\theta(x)$ to be empty). This will lead, respectively, to an infeasible or unbounded equivalent problem (3.8). We can avoid this concern by adjusting the PICNN architecture to ensure its sublevel sets are compact and by suitably increasing q when needed to ensure $\Omega_\theta(x)$ is never empty. Such modifications do not change the general form of the problem (3.8) and preserve the marginal coverage guarantee for the uncertainty set $\Omega_\theta(x)$; see Section 3.9 for details.

3.5 Experiments

In this section, we present experimental results for our E2E method against several ETO baselines. Code to reproduce our results is available on GitHub.[4]

Problem descriptions

We consider two tasks: price forecasting for battery storage operation and portfolio optimization. Their task loss functions and constraints satisfy Assumption 8.

Price forecasting for battery storage. This problem comes from Donti et al. [73], where a grid-scale battery operator predicts electricity prices $y \in \mathbb{R}^T$ over a T-step horizon and uses the predicted prices to decide a battery charge/discharge schedule for price arbitrage. The input features x include the past day's prices and temperature, the next day's energy load forecast and temperature forecast, binary indicators of weekends or holidays, and yearly sinusoidal features. The operator decides how much to charge ($z^{\text{in}} \in \mathbb{R}^T$) or discharge ($z^{\text{out}} \in \mathbb{R}^T$) the battery, which changes the battery's state of charge ($z^{\text{state}} \in \mathbb{R}^T$). The battery has capacity B, charging efficiency γ, and maximum charging/discharging rates c^{in} and c^{out}. The task loss function represents the multiple objectives of maximizing profit, flexibility to participate in other markets by keeping the battery near half its capacity (with weight λ), and battery health by discouraging rapid charging/discharging (with weight ϵ):

$$f(y,z) = \sum_{t=1}^{T} y_t (z^{\text{in}} - z^{\text{out}})_t + \lambda \left\| z^{\text{state}} - \frac{B}{2}\mathbf{1} \right\|^2 + \epsilon \left\| z^{\text{in}} \right\|^2 + \epsilon \left\| z^{\text{out}} \right\|^2.$$

The constraints are, for all $t = 1, \ldots, T$,

$$\begin{aligned} &z_0^{\text{state}} = B/2, \quad z_t^{\text{state}} = z_{t-1}^{\text{state}} - z_t^{\text{out}} + \gamma z_t^{\text{in}}, \\ &0 \le z^{\text{in}} \le c^{\text{in}}, \quad 0 \le z^{\text{out}} \le c^{\text{out}}, \quad 0 \le z_t^{\text{state}} \le B. \end{aligned}$$

Following Donti et al. [73], we set $T = 24$, $B = 1$, $\gamma = 0.9$, $c^{\text{in}} = 0.5$, $c^{\text{out}} = 0.2$, $\lambda = 0.1$, and $\epsilon = 0.05$.

Portfolio optimization. We adopt the portfolio optimization setting and synthetic dataset from Chenreddy and Delage [56], where the prediction targets $y \in \mathbb{R}^n$ are the returns of a set of n securities, and the decision $z \in \mathbb{R}^n$ sets portfolio weights. The task loss is $f(y,z) = -y^\top z$, with constraints $z \ge 0$, $\mathbf{1}^\top z = 1$. The synthetic dataset consists of $(x,y) \in \mathbb{R}^2 \times \mathbb{R}^2$ drawn from a mixture of three 4-D multivariate normal

[4] https://github.com/chrisyeh96/e2e-conformal

distributions. We provide data details in Section 3.11 and experimental results in Section 3.7. The results are similar to those for battery storage, except that portfolio optimization is a lower-dimensional and easier problem.

Baseline methods

We implemented several "estimate-then-optimize" (ETO) baselines, listed below, to compare against our end-to-end (E2E) method. These two-stage ETO baselines are trained using task-agnostic losses such as pinball loss or negative log-likelihood (NLL). To ensure a fair comparison against our E2E method, we also apply conformal calibration to each ETO method after training to satisfy coverage.

- **ETO** denotes models with identical neural network architectures to our E2E models, differing only in the loss function during training. The box uncertainty **ETO** model is trained with pinball loss to predict the $\frac{\alpha}{2}$ and $1-\frac{\alpha}{2}$ quantiles. The ellipsoidal uncertainty **ETO** model is trained with a multivariate normal negative log-likelihood loss. For the PICNN **ETO** model, we train s_θ using a negative log-likelihood loss by interpreting s_θ as an energy function—i.e., $\hat{\mathcal{P}}_\theta(y \mid x) \propto \exp(-s_\theta(x,y))$—yielding the loss $\mathrm{NLL}(\theta) = \ln s_\theta(x,y) + \ln Z_\theta(x)$, where $Z_\theta(x) = \int_{\tilde{y}\in\mathbb{R}^n} \exp(-s_\theta(x,\tilde{y}))\ \mathrm{d}\tilde{y}$, following the approach of Lin and Ba [164]. More details of the **ETO** models are given in Section 3.11.
- **ETO-SLL** is our implementation of the box and ellipsoid uncertainty ETO methods from Sun et al. [256]. Unlike **ETO**, **ETO-SLL** first trains a point estimate model (without uncertainty) with mean-squared error loss. Then, **ETO-SLL** box and ellipsoidal uncertainty sets are derived from training a separate quantile regressor using pinball loss to predict the $(1-\alpha)$-quantiles of absolute residuals or ℓ_2-norm of residuals of the point estimate. Unlike **ETO** which can learn ellipsoidal uncertainty sets with different covariance matrices for each input x, the **ETO-SLL** ellipsoidal uncertainty sets all share the same covariance matrix (up to scale).
- **ETO-JC** is our implementation of the ellipsoid uncertainty ETO method by Johnstone and Cox [123]. Like **ETO-SLL**, **ETO-JC** also first trains a point estimate model (without uncertainty) with mean-squared error loss. **ETO-JC** uses the same covariance matrix (with the same scale) for each input x.

Battery storage problem results

Figure 3.3 (top) compares task loss performance for different uncertainty levels ($\alpha \in \{.01, .05, .1, .2\}$) and the different uncertainty set representations for the ETO

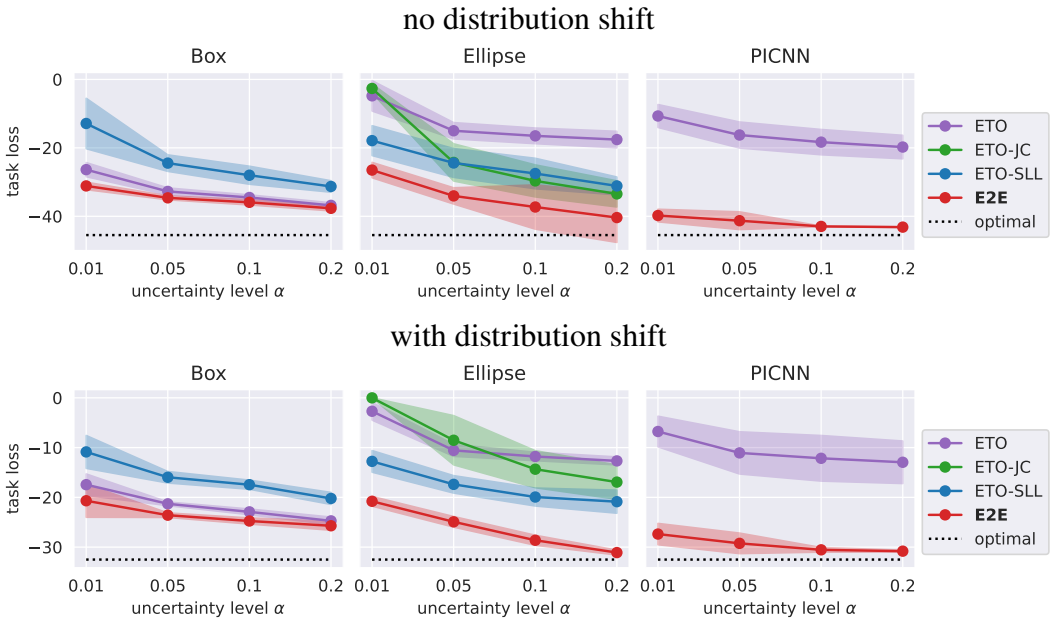


Figure 3.3: Task loss performance (mean ±1 stddev across 10 runs) for the battery storage problem with no distribution shift (top) and with distribution shift (bottom). Lower values are better.

baselines against our proposed E2E methodology on the battery storage problem with no distribution shift. Our E2E approach consistently yields improved performance over all ETO baselines, for all three uncertainty set parametrizations, and over all tested uncertainty levels α. Moreover, the PICNN uncertainty representation, when trained end-to-end, provides up to 42% relative improvement in performance over the best ETO box uncertainty set and up to 209% relative improvement over the best ETO ellipse uncertainty set. We also show the corresponding coverage obtained by the learned uncertainty sets in Figure 3.4 (top); all models obtain coverage close to the target level, confirming that the improvements in task loss performance from our E2E approach do not come at the cost of worse coverage.

In Section 3.7, we present additional results on the battery storage problem, including value-at-risk (VaR) and conditional value-at-risk (CVaR) metrics which show that the E2E approach consistently outperforms the ETO baselines on tail risk as well. In addition, Table 3.1 shows the per-epoch computational time required by our E2E approach compared with the two-stage baselines. While E2E requires additional training time, the training time is dominated by time spent solving the conditional robust optimization problem (3.1) for every training example at every epoch. The development of GPU-accelerated solvers [37] may help significantly speed up the optimization, though we do not explore GPU-accelerated optimization in our

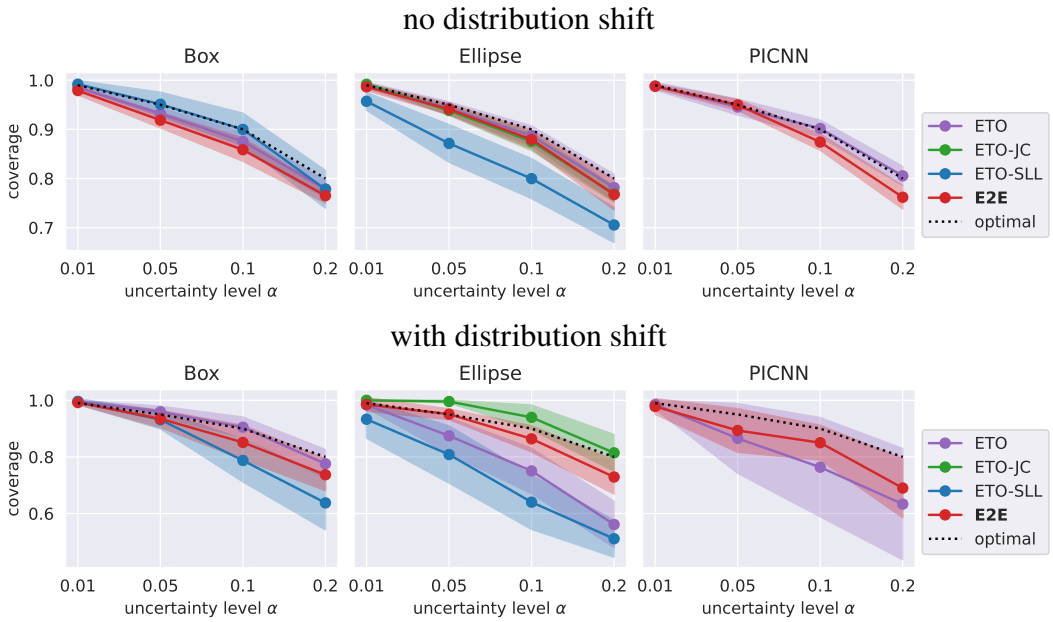


Figure 3.4: Coverage (mean ±1 stddev across 10 runs) for the battery storage problem with no distribution shift (top) and with distribution shift (bottom). The dotted black line indicates the target coverage level $1 - \alpha$. Our E2E models achieve similar coverage to the ETO baselines, confirming that the lower task loss of our E2E models does not come at the expense of worse coverage.

implementation.

Performance under distribution shift

The aforementioned results were produced without distribution shift, where our training and test sets were sampled uniformly at random, thus ensuring exchangeability and guaranteeing marginal coverage. In this section, we evaluate our method on the more realistic setting with distribution shift by splitting our data temporally; our models are trained on the first 80% of days and evaluated on the last 20% of days. Figure 3.8 plots the underlying time series data and visually highlights the distribution shift. Figures 3.3 (bottom) and 3.4 (bottom) mirror Figures 3.3 (top) and 3.4 (top), except that there is now distribution shift. We again find that our E2E approach consistently yields improved performance over all ETO baselines, for all three uncertainty set parametrizations, and for all tested uncertainty levels α. Likewise, the PICNN uncertainties, when trained end-to-end, improve on the performance offered by box and ellipsoidal uncertainty. We unsurprisingly find that, under distribution shift, the models do not provide the same level of coverage guaranteed in the i.i.d. case, as the exchangeability assumption needed for conformal prediction no longer holds. The ellipsoidal and PICNN models tend to provide worse

coverage than the box uncertainty, which we believe reflects how ellipsoidal and PICNN uncertainty sets offer greater representational power, and thus might be fitting too closely to the pre-shift distribution, which impacts robustness under distribution shift. Devising methods to anticipate distribution shift when training these more expressive models, and in particular the PICNN-based uncertainty, remains an interesting avenue for future work.

3.6 Conclusion

In this work, we develop the first end-to-end methodology for training predictive models with uncertainty estimates (with calibration enforced differentiably throughout training) that are utilized in downstream conditional robust optimization problems. We demonstrate an approach utilizing partially input-convex neural networks (PICNNs) to represent general convex uncertainty regions, and we perform extensive experiments on a battery storage application and a portfolio optimization task. Whereas prior works on two-stage estimate-then-optimize approaches emphasized "the convenience brought by the disentanglement of the prediction and the uncertainty calibration" [256], our results highlight that such "convenience" comes at a substantial cost; our end-to-end approach, combined with the expressiveness of the PICNN representation, has clear performance gains over the traditional two-stage methods.

A number of interesting directions for future work on learning decision-aware uncertainty in an end-to-end manner remain. First, while our PICNN-based uncertainty set representation allows the parametrization of general convex uncertainty sets, future work may explore *nonconvex* uncertainty regions. Doing so may require eschewing the analytical methods for differentiating through convex optimization problems and instead use, e.g., policy gradient methods for passing gradients through general stochastic and robust optimization problems. Second, developing end-to-end methods to target conditional calibration (as opposed to marginal calibration) remains an active research direction. Finally, one may explore other types of constraints besides uncertainty set-based robustness, such as value-at-risk (VaR) or conditional value-at-risk (CVaR) constraints.

3.7 Appendix: Additional experimental results

Experimental results: Battery storage

In Figures 3.5 and 3.6, we plot the $(1-\alpha)$-value-at-risk (VaR) and $(1-\alpha)$-conditional value-at-risk (CVaR) [271] of the task loss obtained via our method on the battery

storage problem to evaluate model robustness. In particular, for each target coverage level $1-\alpha$, we evaluate $\text{VaR}^{1-\alpha}$[task loss] and $\text{CVaR}^{1-\alpha}$[task loss] for both the ETO baseline(s) and our E2E methodologies across box, ellipsoid, and PICNN uncertainty sets. In general, it is clear that our E2E methodology uniformly improves over ETO, and while the performance of E2E ellipsoid is close to that of E2E PICNN, the latter appears to perform better for certain α. These results highlight that our methodology improves not only the average task loss, but robustness more generally, even when compared with the robust ETO baselines.

The optimal task losses shown in black dotted lines in Figures 3.3, 3.5 and 3.6 are the lowest achievable task loss on the test set given perfect knowledge of the target y. The optimal task loss is calculated for each example (x, y) in the test set as $f(x, y, z^\star_{\text{opt}})$ where

$$z^\star_{\text{opt}} \;=\; \underset{z\in\mathbb{R}^p}{\arg\min}\, f(x, y, z) \text{ s.t. } g(x, z) \leq 0.$$

Performance. In Table 3.1, we report the wall-clock time for pretraining, optimization, and end-to-end training. These times were measured on a machine with 2× AMD EPYC 7513 32-Core Processors, 1TiB RAM, and 4 NVIDIA A100 GPUs (although only 1 of the GPUs was used in these experiments). The "pretrain" column gives the time per epoch of training with the standard loss function (pinball loss for Box, negative log-likelihood loss for Ellipse and PICNN). The "optimize" column gives the time needed to compute the decision $z^\star_\theta(x)$ given a pretrained model. The "E2E" column gives the time per epoch of E2E training, which requires computing $z^\star_\theta(x)$ for each training and calibration example. Evidently, the bulk of the time spent on E2E training comes from computing the decision $z^\star_\theta(x)$; the additional overhead from computing the gradient through the KKT conditions of the optimization problem is much less than solving the optimization problem itself.

We note that in theory, E2E times should always be higher than the optimization time. The optimization time accounts for the time required to compute $z^\star_\theta(x)$, and E2E training additional accounts for the time required to compute the gradient $\frac{\partial}{\partial\theta} z^\star_\theta(x)$. However, in practice, the optimization time depends heavily on the choice of numerical solver. In our implementation, we use the default `cvxpy` solver (Clarabel) for the optimization step in ETO, whereas we use the default `cvxpylayers` solver (SCS) during E2E training. In the case of the Ellipse and PICNN uncertainty sets, the SCS solver is generally faster than Clarabel; hence, Table 3.1 counterintuitively reports lower E2E training times for Ellipse and PICNN than optimization times.

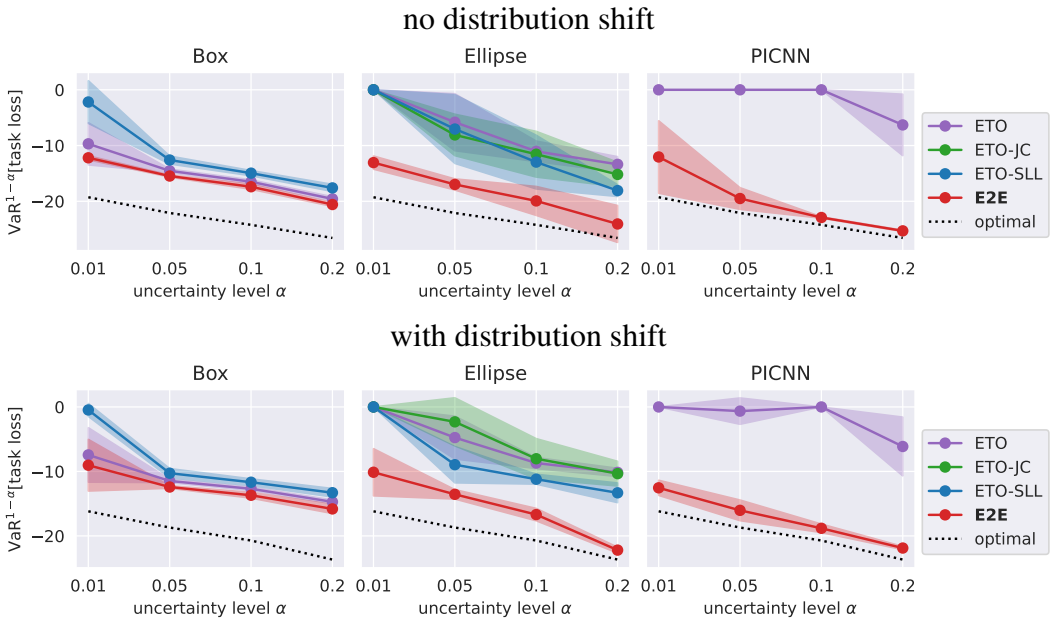


Figure 3.5: Similar to Figure 3.3, except that the value plotted is the $\mathrm{VaR}^{1-\alpha}$[task loss] (mean ±1 stddev across 10 runs) on the test set for the battery storage problem with no distribution shift (top) and with distribution shift (bottom). Lower values are better.

Table 3.1: Time per epoch of pretraining, optimization, and end-to-end training, measured in seconds (wall-clock time). Values are mean ±1 stddev across 10 epochs. Lower values are better.

	pretrain	optimize (train+val)	E2E training
Box	0.03 ± 0.01	4.29 ± 0.07	6.34 ± 0.32
Ellipse	0.03 ± 0.01	7.79 ± 0.06	7.12 ± 1.28
PICNN	4.20 ± 0.20	50.49 ± 0.19	33.24 ± 0.25

We further note that the optimization and E2E training times reflect our particular implementation, but significantly faster times are likely possible. In particular, we solve all of the optimization problems (in both the ETO optimization step and during E2E training) using CPU-based solvers; however, recently developed GPU-accelerated solvers may be able to reduce the optimization time required by orders of magnitude, especially when solving a batch of problems with the same structure [37].

Experimental results: Portfolio optimization

Tables 3.2 and 3.3 show the task loss and coverage results for the portfolio optimization problem. We again find that our E2E approach generally improves upon the ETO baselines at all uncertainty levels α, with the exception of box uncertainty, where all the

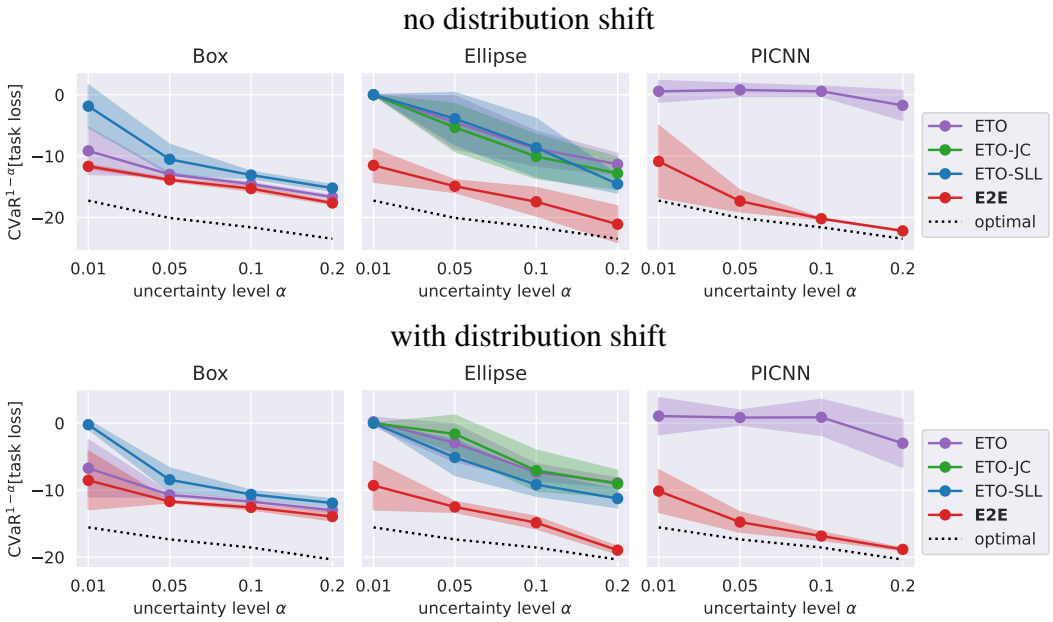


Figure 3.6: Similar to Figure 3.3, except that the value plotted is the $\text{CVaR}^{1-\alpha}$[task loss] (mean ±1 stddev across 10 runs) on the test set for the battery storage problem with no distribution shift (top) and with distribution shift (bottom). Lower values are better.

methods achieve similar performance. Our PICNN-based uncertainty representation, when learned end-to-end, performs better than box uncertainty and comparably with ellipse uncertainty. The similarity in performance between E2E ellipsoidal uncertainty and E2E PICNN uncertainty is likely due to the underlying aleatoric uncertainty (i.e., the uncertainty in $\mathcal{P}(y \mid x)$) generally taking an ellipsoidal shape—the conditional distribution $\mathcal{P}(y \mid x)$ is a Gaussian mixture model, and it tends to have a dominant mode (e.g., see Figure 3.7). In terms of coverage, we find that all the models and training methodologies obtain coverage very close to the target level, confirming that the improvements in task loss performance from our E2E approach do not come at the cost of worse coverage.

Because the conditional distribution $\mathcal{P}(y \mid x)$ for the portfolio optimization problem is 2-dimensional, we can visualize the conditional distribution as well as the uncertainty sets estimated by our models. Figure 3.7 plots the conditional density for input $x = \begin{bmatrix} -1.167 & 0.024 \end{bmatrix}^\top$, along with the $\alpha = 0.1$ uncertainty sets $\Omega_\theta(x)$ and the resulting decision vectors $z^\star_\theta(x)$ for each uncertainty set parametrization. Uncertainty sets and decision vectors from both **ETO** and E2E models are shown in different colors. The key takeaway from this figure is that smaller uncertainty sets (which is what ETO training tends to produce) do not always result in lower task loss. Furthermore,

Table 3.2: Task loss performance (mean ± 1 stddev across 10 runs) for the portfolio optimization problem. Lower values are better, and the best performance for each uncertainty-level α is highlighted. The results show that our E2E methods consistently outperform the ETO baselines.

		uncertainty level α			
		0.01	0.05	0.1	0.2
ETO	Box	-1.16 ± 0.42	-1.37 ± 0.12	-1.39 ± 0.13	-1.41 ± 0.12
ETO	Ellipse	-1.09 ± 0.12	-1.24 ± 0.11	-1.29 ± 0.10	-1.33 ± 0.10
ETO	PICNN	-0.95 ± 0.24	-1.11 ± 0.24	-1.20 ± 0.22	-1.31 ± 0.16
ETO-SLL	Box	-1.41 ± 0.13	-1.42 ± 0.12	-1.42 ± 0.12	-1.44 ± 0.11
ETO-SLL	Ellipse	-1.12 ± 0.22	-1.37 ± 0.12	-1.40 ± 0.12	-1.43 ± 0.12
ETO-JC	Ellipse	-1.16 ± 0.17	-1.40 ± 0.11	-1.42 ± 0.11	-1.44 ± 0.11
E2E	Box	-1.21 ± 0.44	-1.40 ± 0.14	-1.43 ± 0.11	-1.43 ± 0.10
E2E	Ellipse	**-1.48** ± 0.12	-1.47 ± 0.11	**-1.48** ± 0.11	**-1.47** ± 0.11
E2E	PICNN	-1.45 ± 0.14	**-1.48** ± 0.10	**-1.48** ± 0.10	**-1.47** ± 0.11

Table 3.3: Coverage (mean ±1 stddev across 10 runs) for the portfolio optimization problem. Our E2E models achieve similar coverage to the ETO baselines, confirming that the lower task loss of our E2E models does not come at the expense of worse coverage.

		uncertainty level α			
		0.01	0.05	0.1	0.2
ETO	Box	.984 ± .007	.947 ± .017	.902 ± .017	.786 ± .020
ETO	Ellipse	.988 ± .004	.944 ± .020	.894 ± .022	.794 ± .027
ETO	PICNN	.989 ± .006	.949 ± .014	.901 ± .019	.801 ± .034
ETO-SLL	Box	.985 ± .012	.945 ± .021	.885 ± .030	.796 ± .029
ETO-SLL	Ellipse	.989 ± .011	.945 ± .024	.885 ± .039	.795 ± .030
ETO-JC	Ellipse	.991 ± .006	.953 ± .017	.902 ± .026	.796 ± .026
E2E	Box	.989 ± .006	.949 ± .012	.903 ± .016	.785 ± .019
E2E	Ellipse	.992 ± .006	.954 ± .010	.903 ± .022	.798 ± .022
E2E	PICNN	.993 ± .002	.953 ± .010	.912 ± .017	.798 ± .024

the more flexible parametrization of the PICNN allows it to learn uncertainty set shapes that may be more amenable to the downstream robust decision task than box or ellipsoidal uncertainty, even if the resulting uncertainty set has a larger or odder shape.

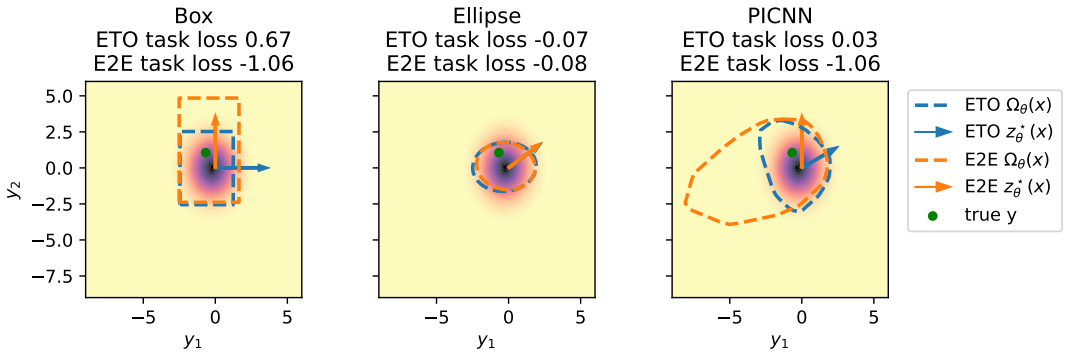


Figure 3.7: This figure plots the density of the conditional distribution $\mathcal{P}(y \mid x)$ for $x = \begin{bmatrix}-1.167 & 0.024\end{bmatrix}^\top$ from the portfolio optimization problem, with darker colors indicating higher density. Also plotted are the $\alpha = 0.1$ uncertainty sets $\Omega_\theta(x)$ (dashed lines) and the resulting decision vectors $z^\star_\theta(x)$ (arrows) for each uncertainty set parametrization. Results for **ETO** models are shown in blue, whereas results for E2E are shown in orange. The "true" y sampled from $\mathcal{P}(y \mid x)$ is drawn in green, and the task loss for this example is computed using this y. The decision vectors have been artificially scaled larger to be easier to see.

3.8 Appendix: Related Work

"Task-based" or "decision-focused" learning. The notion of "task-based" end-to-end model learning was introduced by Donti et al. [73], which proposed to train machine learning models in an end-to-end fashion to minimize a downstream stochastic optimization objective. To achieve this, the authors backpropagate gradients through a stochastic optimization problem, which is made possible for various types of convex optimization problems via the implicit function theorem [73, 7, 3]. However, Donti et al. [73] does not train the model to estimate uncertainty and thereby does not provide any explicit guarantees on robustness on their decisions to uncertainty. Our framework improves upon this baseline by yielding calibrated uncertainty sets which can then be used to obtain robust decisions.

Another line of related works relies on surrogate loss functions to approximate the gradient of the downstream task loss, typically in settings where the exact gradient does not exist or is otherwise hard to compute. For example, the "Smart Predict, then Optimize" approach [78] specifically considers differentiating the solution of linear programs, whereas Wilder et al. [291] differentiates the solution of discrete (combinatorial) optimization problems. As with Donti et al. [73], though, these approaches focus only on average task loss without any estimation of model uncertainty, thereby lacking explicit guarantees on the robustness of the resulting decisions. A recent survey by Mandi et al. [175] provides a comprehensive review of

existing decision-focused learning approaches.

Uncertainty Quantification. Various designs for deep learning regression models that provide uncertainty estimates have been proposed in the literature, including Bayesian neural networks [38, 90], Gaussian process regression and deep kernel learning [218, 292, 166], ensembles of models [144], and quantile regression [222], among other techniques. These methods typically only provide heuristic uncertainty estimates that are not necessarily well-calibrated [183].

Post-hoc methods such as isotonic regression [142] or conformal prediction [233] may be used to calibrate the uncertainty outputs of deep learning models. These calibration methods generally treat the model as a black box and scale predicted uncertainty levels so that they are calibrated on a held-out calibration set. Isotonic regression guarantees calibrated outputs in the limit of infinite data, whereas conformal methods provide probabilistic, finite-sample calibration guarantees when the calibration set is exchangeable (e.g., drawn i.i.d. from the same distribution) with test data. These calibration methods are generally not included in the model training procedure because they involve non-differentiable operators, such as sorting. However, recent works have proposed differentiable losses [77, 254] that approximate the conformal prediction procedure during training and thus allow end-to-end training of models to output more calibrated uncertainty. As approximations, these methods lose the marginal coverage guarantees that true conformal methods provide. However, such guarantees can be recovered at test time by replacing the approximations with true conformal prediction.

Robust and stochastic optimization. The optimization community has proposed a number of techniques over the years to improve robust decision-making under uncertainty, including stochastic, risk-sensitive, chance-constrained, distributionally robust, and robust optimization (e.g., Ben-Tal et al. [22], Shapiro et al. [236], Nemirovski and Shapiro [185], and Rahimian and Mehrotra [216]). These techniques have been applied to a wide range of applications, including energy systems operation [324, 184, 76, 326, 205, 289, 31, 57] and portfolio optimization [99, 213, 29]. In these works, the robust and stochastic optimization methods enable selecting decisions (grid resource dispatches or portfolio allocations) in a manner that is aware of uncertainty, e.g., so an energy system operator can ensure that sufficient generation is available to meet demand even on a cloudy day without much solar generation. Typically, however, the construction of uncertainty sets, estimated probability distributions

over uncertain parameters, or ambiguity sets over distributions takes place offline and is unconnected to the eventual decision-making task. Thus, our proposed end-to-end approach allows for simultaneous calibration of uncertainty sets with optimal decision-making.

3.9 Appendix: Maximizing over the uncertainty set

We consider robust optimization problems of the form

$$\min_{z\in\mathbb{R}^p} \max_{\hat{y}\in\mathbb{R}^n} \ \hat{y}^\top F z + \tilde{f}(x,z) \qquad \text{s.t.} \qquad \hat{y}\in\Omega(x), \quad g(x,z)\leq 0.$$

For fixed z, the inner maximization problem is

$$\max_{\hat{y}\in\mathbb{R}^n} \ \hat{y}^\top F z \qquad \text{s.t.} \qquad \hat{y}\in\Omega(x),$$

which we analyze in the more abstract form

$$\max_{y\in\mathbb{R}^n} c^\top y \qquad \text{s.t.} \qquad y\in\Omega$$

for arbitrary $c\in\mathbb{R}^n\setminus\{0\}$. The subsections of this appendix derive the dual form of this maximization problem for specific representations of the uncertainty set Ω.

Suppose y is standardized or whitened by an affine transformation with $\mu\in\mathbb{R}^n$ and invertible matrix $W\in\mathbb{R}^{n\times n}$

$$y_{\text{transformed}} = W^{-1}(y-\mu)$$

so that Ω is an uncertainty set on the transformed $y_{\text{transformed}}$. Then, the original primal objective can be recovered as

$$c^\top y = c^\top(W y_{\text{transformed}} + \mu) = (Wc)^\top y_{\text{transformed}} + c^\top\mu.$$

In our experiments, we use element-wise standardization of y by setting $W = \operatorname{diag}(y_{\text{std}})$, where $y_{\text{std}}\in\mathbb{R}^n$ is the element-wise standard-deviation of y.

Maximizing over a box constraint

Let $[\underline{y},\overline{y}]\subset\mathbb{R}^n$ be a box uncertainty set for $y\in\mathbb{R}^n$. Then, for any vector $c\in\mathbb{R}^n$, the primal linear program

$$\max_{y\in\mathbb{R}^n} \ c^\top y \qquad \text{s.t.} \qquad \underline{y}\leq y\leq\overline{y}$$

has dual problem

$$\min_{\nu\in\mathbb{R}^{2n}} \ \begin{bmatrix}\overline{y}^\top & -\underline{y}^\top\end{bmatrix}\nu \qquad \text{s.t.} \qquad \begin{bmatrix}I_n & -I_n\end{bmatrix}\nu = c, \quad \nu\geq\mathbf{0},$$

which can also be equivalently written as

$$\min_{\nu \in \mathbb{R}^n} \ (\overline{y} - \underline{y})^\top \nu + \underline{y}^\top c \qquad \text{s.t.} \qquad \nu \geq \mathbf{0}, \quad \nu - c \geq \mathbf{0}.$$

Since strong duality always holds for linear programs, the optimal values of the primal and dual problems will be equal so long as one of the problems is feasible, e.g., so long as the box $[\underline{y}, \overline{y}]$ is nonempty. We can thus incorporate this dual problem into the outer minimization of (3.1) to yield the non-robust form (3.4).

Maximizing over an ellipsoid

For any $c \in \mathbb{R}^n \setminus \{0\}$, $\Sigma \in \mathbb{S}^n_{++}$, and $q > 0$, the primal quadratically constrained linear program (QCLP)

$$\max_{y \in \mathbb{R}^n} \ c^\top y \qquad \text{s.t.} \qquad (y - \mu)^\top \Sigma^{-1} (y - \mu) \leq q$$

has dual problem

$$\min_{\nu \in \mathbb{R}} \ \frac{1}{4\nu} c^\top \Sigma c + \mu^\top c + \nu q \qquad \text{s.t.} \quad \nu \geq 0.$$

By Slater's condition, strong duality holds by virtue of the assumption that $q > 0$ (which implies strict feasibility of the primal problem), and thus the primal and dual problems have the same optimal value. Moreover, since Σ is positive definite and $q > 0$, this problem has a unique optimal solution at $\nu^\star = \frac{1}{2\sqrt{q}} \|L^\top c\|_2$, where L is the unique lower-triangular Cholesky factor of Σ (i.e., $\Sigma = LL^\top$). Substituting $\nu^\star$ into the dual problem yields

$$\sqrt{q} \left\| L^\top c \right\|_2 + \mu^\top c.$$

Plugging this into (3.1) yields the non-robust form (3.5).

We write the dual objective in terms of the Cholesky factor L because our predictive models for ellipsoidal uncertainty directly output the entries of L (see Section 3.11). Note, however, that the dual problem solution can be equivalently written in terms of the square-root of Σ, because

$$\left\| L^\top c \right\|_2^2 = c^\top L L^\top c = c^\top \Sigma c = c^\top \Sigma^{1/2} \Sigma^{1/2} c = \left\| \Sigma^{1/2} c \right\|_2^2.$$

Proof of Theorem 4: Maximizing over the sublevel set of a PICNN

Let $s_\theta : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ be a partially input-convex neural network (PICNN) with ReLU activations as described in (3.7), so that $s_\theta(x, y)$ is convex in y. Suppose that

all the hidden layers have the same dimension d (i.e., $\forall l = 0, \ldots, L-1$: $W_l \in \mathbb{R}^{d \times d}$, $V_l \in \mathbb{R}^{d \times n}$, $b_l \in \mathbb{R}^d$), and the final layer L has $W_L \in \mathbb{R}^{1 \times d}$, $V_L \in \mathbb{R}^{1 \times n}$, $b_L \in \mathbb{R}$. Let $c \in \mathbb{R}^n$ be any vector. Then, the optimization problem

$$\max_{y \in \mathbb{R}^n} \; c^\top y \qquad \text{s.t.} \qquad s_\theta(x, y) \leq q \tag{3.9}$$

can be equivalently written as

$$\max_{y \in \mathbb{R}^n,\ \sigma_1, \ldots, \sigma_L \in \mathbb{R}^d} \quad c^\top y \tag{3.10a}$$

$$\text{s.t.} \quad \sigma_l \geq \mathbf{0}_d \qquad \forall l = 1, \ldots, L \tag{3.10b}$$

$$\sigma_{l+1} \geq W_l \sigma_l + V_l y + b_l \qquad \forall l = 0, \ldots, L-1 \tag{3.10c}$$

$$W_L \sigma_L + V_L y + b_L \leq q. \tag{3.10d}$$

To see that this is the case, first note that (3.10) is a relaxed form of (3.9), obtained by replacing the equalities $\sigma_{l+1} = \text{ReLU}\,(W_l \sigma_l + V_l y + b_l)$ in the definition of the PICNN (3.7) with the two separate inequalities $\sigma_{l+1} \geq \mathbf{0}_d$ and $\sigma_{l+1} \geq W_l \sigma_l + V_l y + b_l$ for each $l = 0, \ldots, L-1$. As such, the optimal value of (3.10) is no less than that of (3.9). However, given an optimal solution $y, \sigma_1, \ldots, \sigma_L$ to (3.10), it is possible to obtain another feasible solution $y, \hat{\sigma}_1, \ldots, \hat{\sigma}_L$ with the same optimal objective value by iteratively decreasing each component of σ_l until one of the two inequality constraints (3.10b), (3.10c) is tight, beginning at $l = 1$ and incrementing l once all entries of σ_l cannot be decreased further. This procedure of decreasing the entries in each σ_l will maintain problem feasibility, since the weight matrices W_l are all assumed to be entrywise nonnegative in the PICNN construction; in particular, this procedure will not increase the left-hand side of (3.10d). Moreover, since one of the two constraints (3.10b), (3.10c) will hold for each entry of each $\hat{\sigma}_l$, this immediately implies that y is feasible for the unrelaxed problem (3.9), and so (3.9) and (3.10) must have the same optimal value.

Having shown that we may replace the convex program (3.9) with a linear equivalent (3.10), we can write this latter problem in the matrix form

$$\max_{y \in \mathbb{R}^n,\ \sigma_1, \ldots, \sigma_L \in \mathbb{R}^d} \; c^\top y \qquad \text{s.t.} \qquad A \begin{bmatrix} y \\ \sigma_1 \\ \vdots \\ \sigma_L \end{bmatrix} \leq b$$

where

$$A = \begin{bmatrix} & -I_d & & \\ & & \ddots & \\ & & & -I_d \\ V_0 & -I_d & & \\ \vdots & W_1 & \ddots & \\ \vdots & & \ddots & -I_d \\ V_L & & & W_L \end{bmatrix} \in \mathbb{R}^{(2Ld+1)\times(n+Ld)}, \qquad b = \begin{bmatrix} \mathbf{0}_d \\ \vdots \\ \mathbf{0}_d \\ -b_0 \\ \vdots \\ -b_{L-1} \\ q - b_L \end{bmatrix} \in \mathbb{R}^{2Ld+1}. \tag{3.11}$$

By strong duality, if this linear program has an optimal solution, its optimal value is equal to the optimal value of its dual problem:

$$\min_{\nu \in \mathbb{R}^{2Ld+1}} b^\top \nu \qquad \text{s.t.} \qquad A^\top \nu = \begin{bmatrix} c \\ \mathbf{0}_{Ld} \end{bmatrix}, \quad \nu \geq 0. \tag{3.12}$$

We can incorporate this dual problem (3.12) into the outer minimization of (3.1) to yield the non-robust form (3.8). For a more interpretable form of this dual problem, let $\nu^{(i)}$ denote the portion of the dual vector ν corresponding to the i-th block-row of matrix A, indexed from 0. That is, $\nu^{(i)} = \nu_{id+1:(i+1)d}$ for $i = 0, \ldots, 2L-1$. Furthermore, let $\mu = \nu_{2Ld+1}$ be the last entry of ν. Written out, the dual problem (3.12) becomes

$$\begin{aligned} \min_{\nu^{(0)},\ldots,\nu^{(2L-1)} \in \mathbb{R}^d,\, \mu \in \mathbb{R}} \quad & \mu(q - b_L) - \sum_{l=0}^{L} b_l^\top \nu^{(L+l)} \\ \text{s.t.} \quad & \begin{bmatrix} V_0^\top & \cdots & V_L^\top \end{bmatrix} \nu_{Ld+1:} = c \\ & W_{l+1}^\top \nu^{(L+l+1)} - \nu^{(L+l)} - \nu^{(l)} = \mathbf{0}_d \qquad \forall l = 0, \ldots, L-1 \\ & \nu \geq 0. \end{aligned}$$

Ensuring feasibility of the PICNN maximization problem

As noted at the end of Section 3.4, it may sometimes be the case that the inner maximization problem of (3.1) is unbounded or infeasible when $\Omega_\theta(x)$ is parametrized by a PICNN, since in general, the sublevel sets of the PICNN might be unbounded, or the q selected by the split conformal procedure detailed in Section 3.3 may be sufficiently small that $\Omega_\theta(x) = \{\hat{y} \in \mathbb{R}^n \mid s_\theta(x, \hat{y}) \leq q\}$ is empty for certain inputs x. We can address each of these concerns using separate techniques.

Ensuring compact sublevel sets. To ensure that the PICNN-parametrized score function $s_\theta(x, y)$ has compact sublevel sets in y, we can redefine the output layer by

setting $V_L = \mathbf{0}_{1\times n}$ and adding a small ℓ^∞ norm term penalizing growth in y:

$$s_\theta(x, y) = W_L\sigma_L + \epsilon\|y\|_\infty + b_L, \tag{3.13}$$

where $\epsilon \geq 0$ is a small penalty term, and where all the remaining parameters and layers remain identical to their definition in (3.7). This modification ensures that, for any fixed x, $s_\theta(x, y)$ has compact sublevel sets, since $\sigma_L \geq 0$ by construction (3.7) and the penalty term $\epsilon\|y\|_\infty$ will grow unboundedly large as y goes to infinity in any direction, so long as $\epsilon > 0$. Moreover, so long as ϵ is chosen sufficiently small and the PICNN is sufficiently deep, this modification should not negatively impact the ability of the PICNN to represent general compact convex uncertainty sets.

Using this modified PICNN (3.13), the maximization problem (3.9) can be written as an equivalent linear program

$$\begin{aligned}
\max_{\substack{y\in\mathbb{R}^n,\ \sigma_1,\ldots,\sigma_L\in\mathbb{R}^d,\\ \kappa\in\mathbb{R}}} \quad & c^\top y && && \text{(3.14a)}\\
\text{s.t.} \quad & \sigma_l \geq \mathbf{0}_d && \forall l = 1,\ldots,L && \text{(3.14b)}\\
& \sigma_{l+1} \geq W_l\sigma_l + V_l y + b_l && \forall l = 0,\ldots,L-1 && \text{(3.14c)}\\
& \kappa \geq y_i,\ \kappa \geq -y_i && \forall i = 1,\ldots,n && \text{(3.14d)}\\
& W_L\sigma_L + \epsilon\kappa + b_L \leq q && && \text{(3.14e)}
\end{aligned}$$

where the equivalence between (3.9) and (3.14) follows the same argument as that employed in the previous section when showing the equivalence of (3.9) and (3.10). We can thus likewise apply strong duality to obtain an equivalent minimization form of the problem (3.14) and incorporate this into the outer minimization of (3.1) to yield a non-robust problem of the general form (3.8).

Ensuring $\Omega_\theta(x)$ is nonempty. If the q chosen by the split conformal procedure is too small such that $\Omega_\theta(x)$ is empty, i.e., $q \leq q_{\min}$ where

$$q_{\min} := \min_{\hat{y}\in\mathbb{R}^n} s_\theta(x, \hat{y}), \tag{3.15}$$

then we simply increase q to $q_{\min}$ so that $\Omega_\theta(x)$ is guaranteed to be nonempty. That is, for input x, we set

$$q = \max\left(\min_{\hat{y}\in\mathbb{R}^n} s_\theta(x, \hat{y}),\ \textsc{Quantile}(\{s_\theta(x_i, y_i)\}_{(x_i,y_i)\in D_{\text{cal}}},\ 1-\alpha)\right).$$

This preserves the marginal coverage guarantee, as increasing q can only result in a larger uncertainty set $\Omega_\theta(x)$.

In theory, $q_{\min}$ varies as a function of θ, and it is possible to differentiate through the optimization problem (3.15) using the methods from Agrawal et al. [3] since the problem is convex and s_θ is assumed to be differentiable w.r.t. θ almost everywhere. However, to avoid this added complexity, in practice, we treat $q_{\min}$ as a constant. In other words, on inputs x where we have to increase q to $q_{\min}$, we treat $\frac{\partial q}{\partial \theta} = 0$.

3.10 Appendix: Exact differentiable conformal prediction

In this section, we prove how to exactly differentiate through the conformal prediction procedure, unlike the approximate derivative first introduced in [254].

Theorem 5. *Let $\alpha \in (0, 1)$ be a risk level, and let $s_i := s_\theta(x_i, y_i)$ denote the scores computed by a score function $s_\theta : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ over data points $\{(x_i, y_i)\}_{i=1}^M$. Suppose $s_\theta(x_i, y_i)$ is differentiable w.r.t. θ for all $i = 1, \ldots, M$.*

Define $s_{M+1} := \infty$. Let $\sigma : \{1, \ldots, M+1\} \to \{1, \ldots, M+1\}$ denote the permutation that sorts the scores in ascending order, such that $s_{\sigma(i)} \leq s_{\sigma(j)}$ for all $i < j$. For simplicity of notation, we may write $s_{(i)} := s_{\sigma(i)}$.

Let $q = \textsc{Quantile}(\{s_i\}_{i=1}^M, \ 1 - \alpha)$ where the $\textsc{Quantile}$ function is as defined in Algorithm 4. That is, $q = s_{(k)}$, where $k := \lceil (M + 1)(1 - \alpha) \rceil \in \{1, \ldots, M, M + 1\}$. If $s_{(k)}$ is unique, then

$$\frac{\mathrm{d}q}{\mathrm{d}\theta} = \begin{cases} \frac{\mathrm{d}}{\mathrm{d}\theta} s_\theta(x_{\sigma(k)}, y_{\sigma(k)}), & \text{if } \alpha \geq \frac{1}{M+1} \\ 0, & \text{otherwise.} \end{cases}$$

Proof. First, when $\alpha \in (0, \frac{1}{M+1})$, we have $k = M + 1$, so $q = \infty$ is constant regardless of the choice of θ. Thus, $\frac{\mathrm{d}q}{\mathrm{d}\theta} = 0$.

Now, suppose $\alpha \geq \frac{1}{M+1}$. The $\textsc{Quantile}$ function returns the k-th largest value of $\{s_i\}_{i=1}^M \cup \{\infty\}$. Since we assume $s_{(k)}$ is unique, we have $\frac{\mathrm{d}q}{\mathrm{d}s_{(i)}} = \mathbf{1}[i = k]$. Finally, we have

$$\frac{\mathrm{d}q}{\mathrm{d}\theta} = \sum_{i=1}^M \frac{\mathrm{d}q}{\mathrm{d}s_i} \frac{\mathrm{d}s_i}{\mathrm{d}\theta} = \sum_{i=1}^M \frac{\mathrm{d}q}{\mathrm{d}s_{(i)}} \frac{\mathrm{d}s_{(i)}}{\mathrm{d}\theta} = \frac{\mathrm{d}s_{(k)}}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta} s_\theta(x_{\sigma(k)}, y_{\sigma(k)}).$$

□

The two key assumptions in this theorem are that (1) s_θ is differentiable w.r.t. θ, and (2) $s_{(k)}$ is unique. When s_θ is a neural network with a common activation function (e.g., ReLU), (1) holds for inputs $(x, y) \in \mathbb{R}^m \times \mathbb{R}^n$ almost everywhere and θ almost

everywhere. Regarding (2), in practice, just as the gradient of the max function is typically implemented without checking whether its inputs have ties, we do not check whether $s_{(k)}$ is unique.

3.11 Appendix: Experiment details

Our experiments were conducted across a variety of machines, including private servers and Amazon AWS EC2 instances, ranging from 12-core to 128-core machines. Our ETO experiments benefited from GPU acceleration across a combination of NVIDIA GeForce GTX 1080 Ti, Titan RTX, T4, and A100 GPUs. Our E2E experiments did not use GPU acceleration, due to the lack of GPU support in the `cvxpylayers` Python package [3].

In all experiments, we use a batch size of 256 and the Adam optimizer [133]. Models were trained for up to 100 epochs with early stopping if there was no improvement in validation loss for 10 consecutive epochs.

For box and ellipsoid **ETO** baseline models, we performed a hyperparameter grid search over learning rates ($10^{-4.5}$, 10^{-4}, $10^{-3.5}$, 10^{-3}, $10^{-2.5}$, 10^{-2}, $10^{-1.5}$) and L2 weight decay values (0, 10^{-4}, 10^{-3}, 10^{-2}). For PICNN **ETO** models we performed a hyperparameter grid search over learning rates (10^{-4}, 10^{-3}, 10^{-2}) and L2 weight decay values (10^{-4}, 10^{-3}, 10^{-2}).

Uncertainty representation

Box uncertainty. Our box uncertainty model uses a neural network h_θ with 3 hidden layers of 256 units each and ReLU activations with batch-normalization. The output layer has dimension $2n$, where dimensions $1:n$ predict the lower bound. Output dimensions $n+1:2n$, after passing through a softplus to ensure positivity, represents the difference between the upper and lower bounds. That is,

$$\begin{bmatrix} h_\theta^{\text{lo}}(x) \\ h_\theta^{\text{hi}}(x) \end{bmatrix} = \begin{bmatrix} h_\theta(x)_{1:n} \\ h_\theta(x)_{1:n} + \text{softplus}(h_\theta(x)_{n+1:2n}) \end{bmatrix}.$$

This architecture ensures that $h_\theta^{\text{hi}}(x) > h_\theta^{\text{lo}}(x)$.

In the two-stage **ETO** baseline, we first train h_θ to estimate the $\alpha/2$- and $(1-\alpha/2)$-quantiles, so that $[h_\theta^{\text{lo}}(x), h_\theta^{\text{hi}}(x)]$ represents the centered $(1-\alpha)$-confidence region. Quantile regression is a common method for generating uncertainty sets for scalar predictions by estimating quantiles of the conditional distribution $\mathcal{P}(y \mid x)$ [222]. For scalar true label y, quantile regression models are commonly trained to minimize

pinball loss (a.k.a. *quantile loss*) where β is the quantile level being estimated:

$$\text{pinball}_\beta(\hat{y}, y) = \begin{cases} \beta \cdot (y - \hat{y}), & \text{if } y > \hat{y} \\ (1 - \beta) \cdot (\hat{y} - y), & \text{if } y \leq \hat{y}. \end{cases}$$

To generalize the pinball loss to our setting of multi-dimensional $y \in \mathbb{R}^n$, we sum the pinball loss across the dimensions of y: $\text{pinball}_\beta(\hat{y}, y) = \sum_{i=1}^{n} \text{pinball}_\beta(\hat{y}_i, y_i)$.

Our end-to-end (E2E) box uncertainty models use the same architecture as above, initialized with weights from the trained **ETO** model. We found it helpful to use a weighted combination of the task loss and pinball loss during training of the E2E models to improve training stability. In our experiments, we used a weight of 0.9 on the task loss and 0.1 on the pinball loss. The E2E models used the best L2 weight decay from the **ETO** models, and the learning rate was tuned across 10^{-2}, 10^{-3}, and 10^{-4}.

Ellipsoidal uncertainty. Our ellipsoidal uncertainty model uses a neural network h_θ with 3 hidden layers of 256 units each and ReLU activations with batch-normalization. The output layer has dimension $n + n(n+1)/2$, where dimensions $1 : n$ predict the mean $\mu_\theta(x)$ and the remaining output dimensions are used to construct a lower-triangular Cholesky factor $L_\theta(x)$ of the covariance matrix $\Sigma_\theta(x) = L_\theta(x)L_\theta(x)^\top$. We pass the diagonal entries of $L_\theta(x)$ through a softplus function to ensure strict positivity, which then ensures $\Sigma_\theta(x)$ is positive definite.

For the **ETO** baseline, we trained the model using the negative log-likelihood (NLL) loss

$$\text{NLL}(\theta) = \frac{1}{N} \sum_{(x,y) \in D} -\ln \mathcal{N}(y \mid \mu_\theta(x), \Sigma_\theta(x)),$$

where $\mathcal{N}(\cdot \mid \mu, \Sigma)$ denotes the density of a multivariate normal distribution with mean μ and covariance matrix Σ.

Our end-to-end (E2E) ellipsoidal uncertainty models use the same architecture as above, initialized with weights from the trained **ETO** model. We found it helpful to use a weighted combination of the task loss and NLL loss during training of the E2E models to improve training stability. In our experiments, we used a weight of 0.9 on the task loss and 0.1 on the NLL loss. The E2E models used the best L2 weight decay from the **ETO** models, and the learning rate was tuned across 10^{-2}, 10^{-3}, and 10^{-4}.

PICNN uncertainty. Our PICNN has 2 hidden layers with ReLU activations.

For the battery storage problem, we used 64 units per hidden layer. We did not run into any feasibility issues for the PICNN maximization problem, so we did not restrict V_L as described in Section 3.9, and we set $\epsilon = 0$.

For the portfolio optimization problem, we tried 32, 64, and 128 units per hidden layer, finding that 32 units worked best. We did run into feasibility issues for the PICNN maximization problem, which we resolved by setting $V_L = \mathbf{0}_{1\times n}$ as described in Section 3.9. This change alone was sufficient, and we set $\epsilon = 0$.

For the **ETO** baseline, we take inspiration from the approach by Lin and Ba [164] to give probabilistic interpretation to a PICNN model s_θ via the energy-based model $\hat{\mathcal{P}}_\theta(y \mid x) = \frac{1}{Z_\theta(x)} \exp(-s_\theta(x, y))$ where $Z_\theta(x) := \int_{\tilde{y}\in\mathbb{R}^n} \exp(-s_\theta(x, \tilde{y}))\, \mathrm{d}\tilde{y}$ is the normalizing constant. We train our **ETO** PICNN models with an approximation to the true NLL loss based on samples from the Metropolis-Adjusted Langevin Algorithm (MALA), a Markov Chain Monte Carlo (MCMC) method. We refer readers to our code for the specific hyperparameters and implementation details we used.

Note that under this energy-based model, adding a scalar constant c to the PICNN (i.e., $s_\theta(x, y) + c$) does not change the probability distribution. That is, $\exp(-s_\theta(x, y)) \propto \exp(-s_\theta(x, y) + c)$. To regularize the PICNN model, which has a bias term in its output layer, we therefore introduce a regularization loss of $w_{\text{zero}} \cdot s_\theta(x, y)^2$ where w_{zero} is a regularization weight. This regularization loss encourages $s_\theta(x, y)$ to be close to 0, for all examples in the training set. In our experiments, we set $w_{\text{zero}} = 1$.

Our end-to-end (E2E) PICNN uncertainty models use the same architecture as above, initialized with weights from the trained **ETO** model. Unlike for box and ellipsoidal uncertainty which used a weighted combination of task loss and NLL loss, our E2E PICNN uncertainty models are trained only with the task loss. Similar to the **ETO** PICNN model, we also regularize the E2E PICNN. Here, we add a regularization loss of $w_{\text{q}} \cdot q^2$, where w_{q} is a regularization weight and q is the conformal prediction threshold computed in each minibatch of E2E training. This regularization loss term aims to keep q near 0; without this regularization, we found that q tended to grow dramatically over training epochs with poor task loss. In our experiments, we set $w_{\text{q}} = 0.01$.

The E2E models used the best L2 weight decay from the **ETO** models. For the battery storage problem, we tested learning rates of 10^{-3} and 10^{-4}. For the portfolio optimization problem, we used a learning rate of 5×10^{-3}.

Data

Price forecasting for battery storage. We use the same dataset as Donti et al. [73] in our price forecasting for battery storage problem. In this dataset, the target $y \in \mathbb{R}^{24}$ is the hourly PJM day-ahead system energy price for 2011-2016, for a total of 2189 days. Unlike Donti et al. [73], though, we do not exclude any days whose electricity prices are too high (>500$/MWh). Whereas Donti et al. [73] treated these days as outliers, our conditional robust optimization problem is designed to output robust decisions. For predicting target for a given day, the inputs $x \in \mathbb{R}^{101}$ include the previous day's log-prices, the given day's hourly load forecast, the previous day's hourly temperature, the given day's hourly temperature, and several calendar-based features such as whether the given day is a weekend or a US holiday.

For the setting without distribution shift, we take a random 20% subset of the dataset as the test set; because the test set is selected randomly, it is considered exchangeable with the rest of the dataset. For the setting with distribution shift, we take the chronologically last 20% of the dataset as the test set; because load, electricity prices, and temperature all have distribution shifts over time, the test set is not exchangeable with the rest of the dataset. Figure 3.8 illustrates the data from the distribution shift setting, where the electricity price tends to be lower and has lower variability in the test set compared to the training/validation splits. For each seed, we further use a 80/20 random split of the remaining data for training and calibration.

Portfolio optimization. For the portfolio optimization task, we used synthetically generated data. We sample $x \in \mathbb{R}^2$, $y \in \mathbb{R}^2$ from a mixture of three 4-D multivariate Gaussian distributions as used in [56]. Formally,

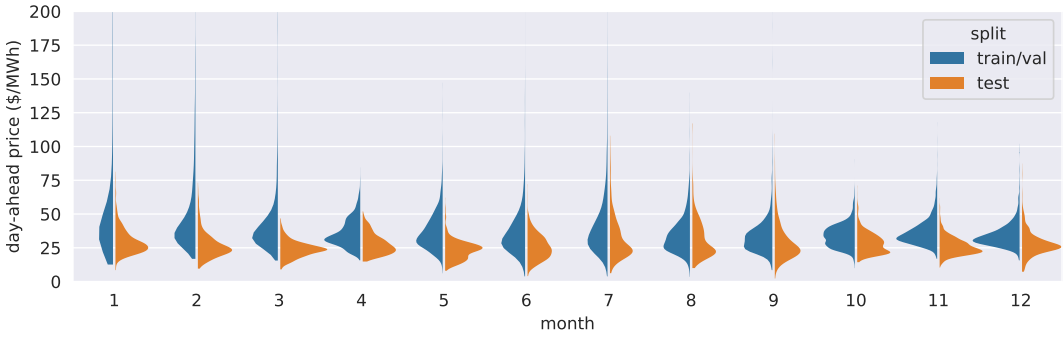
$$\begin{bmatrix} x \\ y \end{bmatrix} \sim p_a \mathcal{N}(\mu_a, \Sigma_a) + p_b \mathcal{N}(\mu_b, \Sigma_b) + p_c \mathcal{N}(\mu_c, \Sigma_c)$$

where $p_a + p_b + p_c = 1$. Specifically,

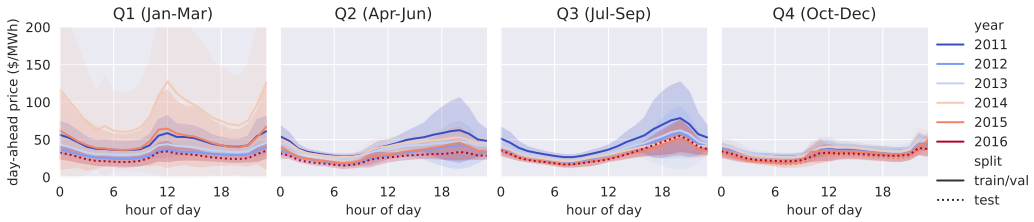
$$p_a = \phi, \qquad p_b = \frac{1}{\alpha_{\text{GMM}} + 1}(1 - \phi), \quad p_c = \frac{\alpha_{\text{GMM}}}{\alpha_{\text{GMM}} + 1}(1 - \phi),$$

$$\mu_a = \mathbf{0}_4, \qquad \mu_b = \begin{bmatrix} 0 & 5 & 5 & 0 \end{bmatrix}^\top, \qquad \mu_c = \mu_b,$$

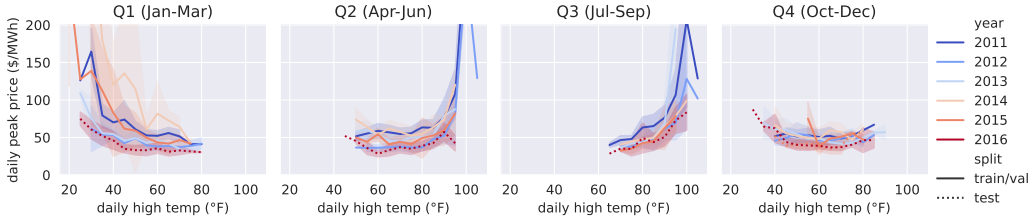
$$\Sigma_a = \begin{bmatrix} 1 & 0 & 0.37 & 0 \\ 0 & 1.5 & 0 & 0 \\ 0.37 & 0 & 2 & 0.73 \\ 0 & 0 & 0.73 & 3 \end{bmatrix}, \quad \Sigma_b = \alpha_{\text{GMM}} \Sigma_a, \qquad \Sigma_c = \frac{1}{\alpha_{\text{GMM}}} \Sigma_a,$$



(a) Month of year vs. electricity price



(b) Hour of day vs. electricity price (shaded region shows ± 1 std dev)



(c) Daily high temperature (rounded to nearest 5°F) vs. daily peak electricity price (shaded region shows ± 1 std dev)

Figure 3.8: Visualization of data from the battery storage problem under distribution shift. The train/val splits are sampled randomly from the range 2011-01-04 to 2015-10-20, whereas the test set comprises 2015-10-21 to 2016-12-31. These plots evidently show that the electricity price is generally lower in the test set and also has lower variability by both hour of day and the daily maximum temperature.

for some $\phi \in [0, 1]$ and $\alpha_{\text{GMM}} \in [0, 1]$. In our experiments, we used $\phi = 0.7$ and $\alpha_{\text{GMM}} = 0.9$. (Chenreddy and Delage [56] do not disclose the values of ϕ and α_{GMM} chosen for their experiments.) For each random seed, we generate 2000 samples and use a (train, calibration, test) split of (600, 400, 1000).

Chapter 4

ROBUST ENERGY STORAGE OPERATION VIA GENERATIVE WASSERSTEIN DISTRIBUTIONALLY ROBUST OPTIMIZATION

Whereas the previous two chapters focused on end-to-end differentiation through conformal risk control guarantees, this chapter and the subsequent chapter eschew formal risk guarantees and instead focus on end-to-end differentiation through decision problems where the uncertainty is represented by a distribution, as opposed to a single point estimate or a set. In particular, this chapter focuses on learning conditional distributions for distributionally robust optimization (DRO) problems, where the "learned uncertainty representation" is a distributional ambiguity set defined by a Wasserstein ball around the learned conditional distribution.

This chapter is primarily based on the following paper:

[295] Han Xu and Christopher Yeh. 2025. Robust energy storage operation via generative Wasserstein distributionally robust optimization. In *NeurIPS 2025 Workshop on Tackling Climate Change with Machine Learning*. San Diego, CA, USA, (Dec. 2025). https://www.climatechange.ai/papers/neurips2025/79.

4.1 Introduction

Machine learning techniques are widely used for predicting uncertain quantities for decision-making under uncertainty in many real-world problems including energy systems [73, 161], finance [61], supply chain management [121], among many other domains. Traditional predict-then-optimize and estimate-then-optimize (ETO) approaches first train predictive models for prediction accuracy, then use predictions in downstream optimization [78]. However, this separation between training and optimization causes misalignment: prediction errors optimized for statistical metrics may severely degrade decision quality which is also known as the "optimizer's curse" [248]. End-to-end (E2E) approaches (also known as decision-focused learning [175]) address this by training models to directly minimize downstream decision costs using differentiable optimization [73].

Recent E2E approaches have explored incorporating risk aversion by predicting an uncertainty set instead of a point estimate for the uncertain parameter, and then solving a robust optimization (RO) problem [56, 281, 311] to minimize the worst-case cost over the learned uncertainty set. However, the E2E-RO approach has two

limitations. First, to ensure tractability, E2E-RO requires the uncertainty set to be convex, which limits the expressivity of the uncertainty representation. Additionally, the worst-case optimization approach can lead to overly conservative solutions [268].

Instead of learning an uncertainty set for the uncertain parameter, an alternative approach is to quantify uncertainty around a learned conditional distribution of the parameter. **In this work, we propose a novel generative Wasserstein distributionally robust optimization framework (Gen-WDRO) that combines generative modeling with distributionally robust optimization for end-to-end learning.** Our approach provides a flexible uncertainty representation, and we demonstrate robustness against distribution shift in experiments on a battery storage optimization problem. By improving battery storage profitability, this approach shows its potential for supporting the deployment of energy storage systems that enable greater renewable energy integration and climate mitigation.

4.2 Related Works

Robust End-to-End Learning. Donti et al. [73] pioneered task-based end-to-end learning, which trains models to minimize decision costs rather than prediction error. Differentiable optimization layers [3] enable tractable gradient computation through convex programs. Yeh* et al. [311] utilize conformal prediction to calibrate uncertainty sets and employ input-convex neural networks to represent general convex uncertainty sets.

Distributionally Robust End-to-End Learning. Costa and Iyengar [61] developed KL-divergence-based distributionally robust portfolio optimization. Ma et al. [171] proposed generic differentiable DRO layers for mixed-integer problems using second-order cone ambiguity sets, mentioning the possibility of using Wasserstein distance to construct ambiguity sets but without experimental validation. Nguyen et al. [186] propose a conditional distributionally robust optimization strategy that solves conditional DRO problems without requiring explicit learning of the conditional distribution. Liang et al. [161] propose using DRO to solve two-stage decision-making problems in an end-to-end manner, though they do not learn conditional distributions.

Generative Models for Decision-Making. Our work is also related to generative modeling which has been applied in various fields such as image generation [112] and drug design [169]. In the context of decision-making, Wang et al. [284] employ

generative models to learn conditional distributions for CVaR optimization. Rather than using robust optimization, they directly optimize the CVaR value as their objective function.

Unlike existing work, our approach integrates generative modeling with DRO for end-to-end learning under distribution shift, providing both flexible conditional distribution learning and enhanced robustness guarantees.

4.3 Problem Formulation and Methodology

Problem Statement

Consider an agent that observes an input $x \in \mathbb{R}^m$ and must make a decision $z \in \mathbb{R}^p$ without knowing the future outcome $y \in \mathbb{R}^n$. Once z is chosen and y is revealed, the agent incurs a task loss $f(x, y, z)$ defined by a loss function $f : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$. We assume that the decision z must adhere to hard constraints $g(x, z) \leq 0$ which do not depend on the future outcome y. The future outcome y is uncertain and only influences the cost function $f(x, y, z)$. Let $\mathbb{P}$ denote the unknown joint distribution of (x, y). The agent's goal is to minimize the expected cost while satisfying the constraints:

$$z^*(x) := \arg\min_z \mathbb{E}^{\mathbb{P}}[f(x, y, z) \mid x] \quad \text{s.t. } g(x, z) \leq 0 \tag{4.1}$$

where $\mathbb{E}^{\mathbb{P}}[\cdot \mid x]$ denotes conditional expectation given x.

The above problem cannot be solved directly since the conditional distribution $\mathbb{P}(y \mid x)$ is not known. Instead, we assume the agent has access to a set of i.i.d. samples $\{(x_i, y_i)\}_{i=1}^N$ from the joint distribution $\mathbb{P}$. Given the samples, we then learn a conditional distribution $\hat{\mathbb{P}}_\theta(\cdot \mid x)$ with learnable parameters θ to estimate the conditional expectation $\mathbb{E}^{\mathbb{P}}[f(x, y, z) \mid x]$. However, the learned conditional distribution may not be accurate, or distribution shifts may occur, leading to poor decision-making performance under the learned distribution. To address this issue, we propose using distributionally robust optimization (DRO) to ensure that the decision is robust against an ambiguity set of distributions $\mathcal{B}_\rho(\hat{\mathbb{P}}_\theta(\cdot \mid x))$ that is centered at the learned conditional distribution $\hat{\mathbb{P}}_\theta(\cdot \mid x)$ with some distribution metric $d(\cdot, \cdot)$:

$$\mathcal{B}_\rho(\hat{\mathbb{P}}_\theta(\cdot \mid x)) := \left\{\mathbb{Q} \in \mathcal{P}(\mathbb{R}^n) : \ d(\mathbb{Q}, \hat{\mathbb{P}}_\theta(\cdot \mid x)) \leq \rho\right\}. \tag{4.2}$$

In this case, the agent's robust decision policy can be formulated as the following

DRO problem:

$$z^*(x) := \arg\min_{z} \max_{\mathbb{Q}\in\mathcal{B}_\rho(\hat{\mathbb{P}}_\theta(\cdot|x))} \mathbb{E}^{\mathbb{Q}}[f(x,y,z)] \quad \text{s.t. } g(x,z) \le 0 \tag{4.3}$$

Ambiguity Set Construction and Problem Tractability

Our challenge is to ensure tractability and differentiability of the robust decision policy (4.3) so that we can learn a conditional distribution $\hat{\mathbb{P}}_\theta(\cdot \mid x)$ from the data samples in an end-to-end way. We model the conditional distribution $\hat{\mathbb{P}}_\theta(\cdot \mid x)$ with a conditional normalizing flow (CNF) [293]. Because computing an ambiguity set directly around a CNF model is intractable, we instead approximate the learned CNF with an empirical distribution. For each input x, we sample M predictions $\{\hat{y}_i\}_{i=1}^M$ from the learned CNF to construct a discrete empirical distribution $\hat{\mathbb{Q}}_M(y) = \frac{1}{M}\sum_{i=1}^M \delta_{\hat{y}_i}$. Then, we construct the ambiguity set (4.2) using the Wasserstein metric

$$d_{\mathrm{W}}(\mathbb{Q}, \hat{\mathbb{Q}}_M) = \inf_{\gamma\in\Gamma(\mathbb{Q},\hat{\mathbb{Q}}_M)} \int_{\mathbb{R}^n\times\mathbb{R}^n} \|y - y'\|\, d\gamma(y,y') \tag{4.4}$$

where $\Gamma(\mathbb{Q}, \hat{\mathbb{Q}}_M)$ is the set of all joint distributions γ on $\mathbb{R}^n \times \mathbb{R}^n$ with marginals $\mathbb{Q}$ and $\hat{\mathbb{Q}}_M$, and $\|\cdot\|$ is any norm on $\mathbb{R}^n$. In our experiments, we use the 2-norm $\|\cdot\|_2$.

The following theorem shows that the DRO problem (4.3) based on this ambiguity set can be reformulated as a tractable convex optimization problem under certain conditions.

Theorem 6. *Suppose the task loss function has the form $f(x,y,z) = \tilde{f}(x,z) + \langle y, Fz\rangle$ where $F \in \mathbb{R}^{n\times p}$, and both $\tilde{f}(x,z)$ and the constraint function $g(x,z)$ are convex in z. Then, problem (4.3) with ambiguity set $\mathcal{B}_\rho(\hat{\mathbb{Q}}_M)$ based on the Wasserstein metric can be reformulated as the convex optimization problem*

$$z^*(x) := \arg\min_{z}\ \rho\|Fz\|_\star + \tilde{f}(x,z) + \frac{1}{M}\sum_{i=1}^M \langle \hat{y}_i, Fz\rangle \quad s.t.\ g(x,z) \le 0 \tag{4.5}$$

where $\|\cdot\|_\star$ is the dual norm of the norm $\|\cdot\|$ used in the definition of the Wasserstein distance (4.4).

The proof of this theorem is a simple application of [179, Corollary 5.1] by reformulating the inner maximization problem in (4.3).

Notice that the choice of the radius ρ is flexible. In our approach, we employ a neural network to learn the radius ρ as a function of the input x. In practice, the uncertainty of the learned distribution can vary depending on the specific input x, so it is important for the radius ρ to adapt accordingly.

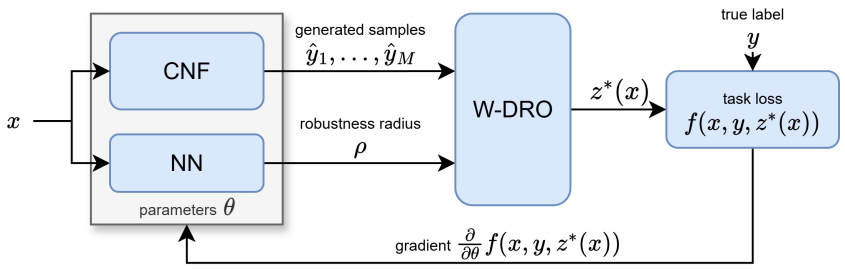


Figure 4.1: Our proposed Gen-WDRO method utilizes a CNF to generate a discrete distribution and a DRO layer to improve robustness. The parameters are updated using gradients from the task loss.

End-to-End Training

Algorithm 5 End-to-end training of the Gen-WDRO method.

function TRAIN(training data $D = \{(x_i, y_i)\}_{i=1}^N$, initial model parameters θ)
  for mini-batch $B \subset \{1, \ldots, N\}$ **do**
    for $i \in B$ **do**
      Compute the gradient of the likelihood loss g_i^{lik}.
      Sample M i.i.d. samples $\{\hat{y}_j\}_{j=1}^M$ from the CNF neural network.
      Solve the convex optimization problem (4.5).
      Compute the gradient of the task loss: $g_i^{\text{task}} = \partial f(x_i, y_i, z^*(x_i))/\partial\theta$.
    Update parameters θ using the gradients $\sum_{i \in B} \alpha_{\text{task}} g_i^{\text{task}} + \alpha_{\text{lik}} g_i^{\text{lik}}$

After reformulating the DRO problem as a convex optimization problem, we propose a methodology in Algorithm 5 to enable end-to-end training of the decision-making policy. For simplicity, the parameters of both the CNF model and the neural network for ρ are collectively denoted as θ. Our end-to-end training approach uses mini-batch gradient descent to minimize a weighted loss function

$$\ell_i(\theta) = -\alpha_{\text{lik}} \log \hat{\mathbb{P}}_\theta(y_i \mid x_i) + \alpha_{\text{task}} f(x_i, y_i, z^*(x_i))$$

which accounts for both the likelihood loss of the CNF neural network and the task loss of the decision-making policy. The gradient of the likelihood loss $g_i^{\text{lik}} := \frac{\partial}{\partial\theta} \log \hat{\mathbb{P}}_\theta(y_i \mid x_i)$ is computed as in [293]. The gradient of the task loss on a single instance is $g_i^{\text{task}} := \frac{\partial f}{\partial z}|_{(x_i, y_i, z^*)} \frac{\partial z^*}{\partial \theta}|_{x_i}$, where $\frac{\partial z^*}{\partial \theta}|_{x_i}$ can be computed by differentiating through the Karush-Kuhn-Tucker (KKT) conditions of the convex reformulated DRO problem (4.5) [3]. Specifically, $\frac{\partial z^*}{\partial \theta}$ can be expressed as $\frac{\partial z^*}{\partial \theta} = \sum_{i=1}^M \frac{\partial z^*}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial \theta} + \frac{\partial z^*}{\partial \rho} \frac{\partial \rho}{\partial \theta}$, and the gradients $\frac{\partial z^*}{\partial \hat{y}_i}$ and $\frac{\partial z^*}{\partial \rho}$ can be computed by differentiating through the convex optimization problem.

4.4 Experimental Results

We consider the problem of grid-scale battery storage participating in day-ahead electricity markets, where the operator seeks to optimally schedule charging and discharging operations for the next day to maximize profit through price arbitrage while satisfying operational constraints. We follow the same setup as in Donti et al. [73]. The input features x include the past day's prices and temperature, the next day's energy load forecast and temperature forecast, binary indicators of weekends or holidays, and yearly sinusoidal features. The operator utilizes these features to predict future energy prices $y \in \mathbb{R}^T$ over a T-step horizon, and decides how much and when to charge ($z^{\text{in}} \in \mathbb{R}^T$) or discharge ($z^{\text{out}} \in \mathbb{R}^T$) the battery, together represented by the decision variable z.

The battery is characterized by its storage capacity B, charging efficiency γ, and maximum charging and discharging rates c_{in} and c_{out}, respectively. The objective function balances multiple goals: profit maximization, and maintaining operational flexibility by keeping the battery state of charge near 50% of capacity (weighted by parameter λ) to enable participation in multiple markets:

$$f(y, z) = \sum_{t=1}^{T} y_t (z_t^{\text{in}} - z_t^{\text{out}}) + \lambda \left\| z_t^{\text{state}} - \frac{B}{2} \right\|^2$$

with constraints given by

$$z_0^{\text{state}} = B/2, \quad z_t^{\text{state}} = z_{t-1}^{\text{state}} - z_t^{\text{out}} + \gamma z_t^{\text{in}} \quad \forall t = 1, \ldots, T$$
$$0 \le z_t^{\text{in}} \le c^{\text{in}}, \quad 0 \le z_t^{\text{out}} \le c^{\text{out}}, \quad 0 \le z_t^{\text{state}} \le B.$$

Following [73], we set $T = 24$, $B = 1$, $\gamma = 0.9$, $c^{\text{in}} = 0.5$, $c^{\text{out}} = 0.2$, and $\lambda = 0.1$.

We set the following parameters for our proposed Gen-WDRO model: $\alpha_{\text{task}} = 0.8$, $\alpha_{\text{lik}} = 0.2$, and $M = 10$.

We evaluate our approach on this problem under distribution shift, where the test set is corrupted with Gaussian $\mathcal{N}(0, 0.3)$ noise added to standardized future electricity prices. This simulates the increased volatility expected from growing renewable energy penetration.

Figure 4.2 compares our Gen-WDRO approach (with trainable radius) against other baselines: ETO-0 and ETO-0.8, representing two-stage estimate-then-optimize approaches using fixed-radius DRO layers with $\rho = 0$ and $\rho = 0.8$ ($\rho = 0.8$ is chosen as the mean radius from our Gen-WDRO model), and fixed-radius variants

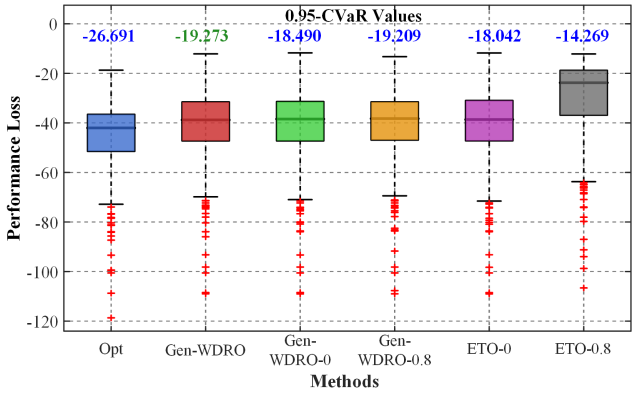


Figure 4.2: Performance comparison on the battery management problem. Each method is tested on 438 test cases, and 0.95-CVaR is adopted for robustness comparison. Lower values are better.

Gen-WDRO-0 and Gen-WDRO-0.8. The "Opt" benchmark represents the optimum obtained using ground-truth prices for optimization.

Our Gen-WDRO method demonstrates superior robustness with the best 0.95-CVaR performance,[1] and outperforms baselines. Additionally, Gen-WDRO achieves more stable performance compared to fixed-radius approaches and substantially outperforms ETO methods. This shows that learning the uncertainty radius end-to-end, rather than fixing it, leads to better performance.

4.5 Conclusion

We have presented a generative Wasserstein distributionally robust optimization (Gen-WDRO) method for end-to-end decision-making under uncertainty. By integrating generative modeling with DRO, our approach enables flexible conditional distribution learning and enhances robustness against distribution shifts. Experimental results demonstrate its potential for helping robust decision making in electricity markets and power systems despite distribution shift, providing a pathway for scaling energy storage deployment to support higher renewable energy penetration and climate goals.

[1]The α-conditional value-at-risk (CVaR) of a random variable X is defined as $\mathbb{E}[X \mid X \geq q_\alpha(X)]$, where $q_\alpha(X)$ is the α-quantile of X [271]. CVaR is a common metric for evaluating robustness [281, 284].

Chapter 5

DIFFUSION-DFL: DECISION-FOCUSED DIFFUSION MODELS FOR STOCHASTIC OPTIMIZATION

This chapter concludes Part I of the thesis by focusing on learning a conditional distribution from which we sample scenarios for stochastic optimization. We can think of this approach as learning "decision-focused scenarios," and the scenarios form our learned uncertainty representation. In particular, we propose to learn the conditional distribution using a diffusion model, which is a powerful generative modeling framework that has achieved state-of-the-art performance in various domains. We then optimize the decision by solving a stochastic optimization problem with samples drawn from the diffusion model. We also propose a lightweight score function estimator that avoids backpropagation through the sampling process, which is memory and compute-intensive. Our approach consistently outperforms strong baselines in decision quality across various tasks.

This chapter is primarily based on the following paper:

[323] Zihao Zhao, Christopher Yeh, Lingkai Kong Kong, and Kai Wang. 2026. Diffusion-DFL: decision-focused diffusion models for stochastic optimization. In *The Fourteenth International Conference on Learning Representations*. (Apr. 2026). https://openreview.net/forum?id=uhv3f80jmG.

5.1 Introduction

Many real-life decision-making tasks require selecting actions that minimize a cost function involving unknown, context-dependent parameters. These parameters must often be predicted from observed features. For example, in supply chain management, future product demand must be estimated before deciding how much inventory to order [263]. A common approach is the predict-then-optimize pipeline, where a predictive model is first trained using a loss function such as mean squared error (MSE), and the resulting predictions are then passed to an optimization solver to guide decisions. While simple and widely adopted, this two-stage method can be misaligned with the true objective: minimizing decision cost. In particular, lower prediction error does not always lead to higher-quality decisions [30, 78].

Decision-focused learning (DFL) addresses this misalignment by integrating the prediction and optimization stages into a single end-to-end framework [73, 291,

175]. Unlike the two-stage approach, DFL trains the prediction model specifically to improve decision outcomes, often resulting in solutions with lower regret. However, most existing DFL methods rely on point (deterministic) predictions as inputs to the optimization layer, despite the fact that in many real-world scenarios, the underlying parameters are inherently uncertain and may follow complex distributions. Ignoring this uncertainty can lead to overconfident models and degraded decision quality [138].

In this work, we introduce a novel DFL approach that leverages diffusion probabilistic models to capture the environment uncertainty in an end-to-end fashion. Here, we use a conditional diffusion model [266] to represent the distribution of uncertain parameters given contextual features. The advantage of integrating a diffusion model into DFL is that, unlike simple distribution predictions (e.g., Gaussian), diffusion models can capture multi-modal or complex distributions. However, the sequential sampling procedure of diffusion models introduces a challenge when training a diffusion model end-to-end for stochastic optimization. To address this, we develop two algorithms: reparameterization and score function. First, the reparameterization trick is a common approach that expresses a random sample as a deterministic function of the model parameters and some noise, and we can backpropagate through sampled prediction to solve the DFL problem.

However, this approach can be very costly in memory and computation because it requires differentiating (and therefore tracking gradients) through the diffusion sampling process. To address this, we introduce a lightweight score function estimator that avoids differentiating through the sampling process. Specifically, we use a score function surrogate to approximate the gradient of the diffusion predictor and plug it into the KKT (Karush-Kuhn-Tucker) implicit-differentiation approach to obtain the total derivative of the decision objective. In addition, we further mitigate the high variance that arises from using only score functions for a few steps by employing a tailored importance sampling strategy. Specifically, instead of sampling diffusion timesteps uniformly, we assign sampling probabilities proportional to the gradient magnitude at each timestep, which keeps the estimator unbiased while substantially reducing its variance.

We evaluate our proposed methods in various applications, including (synthetic) product allocation, energy scheduling, and stock portfolio optimization. Experimental results show that our diffusion DFL methods consistently outperform all baselines, with larger gains on larger-scale problems. Moreover, the score function estimator achieves decision quality comparable to that of the reparameterization method, while

significantly reducing GPU memory usage from 60.75 GB to 0.13 GB.

The contributions of this chapter are the following:

- We introduce the first DFL method that uses diffusion models to capture the downstream uncertainty and employs the reparameterization trick for end-to-end gradient estimation.
- We propose a lightweight score function estimator that avoids backpropagating gradients through the reverse process in the reparameterization method, significantly reducing memory and computation cost.
- We evaluate our methods in three real-world optimization tasks and observe consistent improvements over strong baselines.

5.2 Related Works

Decision-focused learning. DFL is a growing and increasingly influential approach that trains models end-to-end to directly optimize decision quality rather than minimize prediction error [73, 291, 175]. Despite the success in aligning learning objectives with decision-making, a limitation of most existing DFL methods is that they typically rely on **deterministic point predictions** of uncertain parameters [291, 234]. By ignoring distributional uncertainty, deterministic point predictions cannot represent the full outcomes and may lead to lower decision quality [284]. Empirically, classic DFL was observed to struggle in high-dimensional and risk-sensitive real-world settings with significant uncertainty [174].

Therefore, the gap in uncertainty modeling motivates the need for more comprehensive DFL with *stochastic predictions*, where several works have started integrating uncertainty awareness into the DFL pipeline [246, 284, 237, 119]. For instance, [284] proposes a generative DFL approach (Gen-DFL) based on normalizing flow models as the predictor. However, normalizing flows require a bijective network architecture, which restricts the expressiveness of the stochastic predictor.

In this chapter, we propose using diffusion models [112] as a more expressive predictor. By leveraging diffusion models in the DFL paradigm, our approach extends DFL by predicting an accurate full distribution of the unknown parameters, which addresses the overconfidence of deterministic optimization and better aligns with downstream decision-making needs.

Orthogonally, [246] propose using score function gradient estimation to extend DFL to settings with non-differentiable optimization solvers. However, their approach relies on predictors for which the score admits a closed-form expression. In contrast,

the score cannot be computed in closed form for diffusion models. To address this, we approximate the score using the gradient of the ELBO as a surrogate and provide a theoretical bound on the approximation error.

Diffusion models in optimization. Diffusion probabilistic models have achieved great success in modeling high-dimensional data distributions in recent years [249, 252, 69]. Originally popularized for image generation and related structured outputs, their ability to capture multi-modal and high-variety distributions has made them attractive beyond vision tasks, such as combinatorial optimization [258, 229] and black-box optimization [140, 139].

To our best knowledge, however, no prior work has integrated diffusion models into a predict-then-optimize learning pipeline for decision tasks. By using a conditional diffusion model, we can learn a rich distribution over the uncertain inputs and then propagate this uncertainty through to the downstream decision via gradient-based training (score function and reparameterization). This approach combines the strengths of expressive generative modeling and DFL to improve decision quality under uncertainty.

5.3 Problem Statement

Decision-focused learning

We consider a general predict-then-optimize setting [73, 78], where the goal is to make decisions under uncertainty about a key problem parameter. Given a feature vector $x \in \mathcal{X}$ and a prediction of an unknown parameter $y^* \in \mathcal{Y}$, the decision-maker selects $z \in \mathbb{R}^d$ to minimize a decision loss function $f : \mathcal{Y} \times \mathbb{R}^d \to \mathbb{R}$, which measures the cost of applying decision z when the true parameter is y^*. We assume a joint distribution $\mathcal{D}$ over (x, y^*) pairs.

DFL integrates prediction and optimization into a unified framework. The goal is to learn a decision function $z^*_\theta : \mathcal{X} \to \mathbb{R}^d$, parameterized by θ, that minimizes the expected decision loss,

$$\min_\theta F(\theta) := \mathbb{E}_{(x,y^*)\sim\mathcal{D}}[f(y^*, z^*_\theta(x))].$$

The decision $z^*_\theta(x)$ is typically obtained by solving an optimization problem involving a prediction of the uncertainty parameter. Most DFL methods [175] use a deterministic point prediction $y_\theta(x)$ of the uncertain parameter y^*:

$$z^*_\theta(x) = \arg\min_z f(y_\theta(x), z), \quad \text{s.t. } Gz \le h,\ Az = b,$$

where $G \in \mathbb{R}^{n\times d}, h \in \mathbb{R}^n, A \in \mathbb{R}^{p\times d}, b \in \mathbb{R}^p$ are constraint problem coefficients[1].

In contrast, we consider a probabilistic model $P_\theta(\cdot \mid x)$ for the uncertain parameter y^* and let $z^*_\theta(x)$ be the solution to a stochastic optimization problem:

$$z^*_\theta(x) = \arg\min_z \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)], \quad \text{s.t. } Gz \le h,\ Az = b. \tag{5.1}$$

We aim to learn the model parameter θ such that z^*_θ minimizes the expected decision loss $F(\theta)$. By the chain rule, the derivative of F is

$$\frac{\mathrm{d}F(\theta)}{\mathrm{d}\theta} = \mathbb{E}_{(x,y^*)\sim\mathcal{D}}\left[\frac{\partial f(y^*, z^*_\theta(x))}{\partial z}\frac{\mathrm{d}z^*_\theta(x)}{\mathrm{d}\theta}\right].$$

However, computing this gradient (specifically, the $\frac{\mathrm{d}z^*_\theta}{\mathrm{d}\theta}$ term) is challenging because z^*_θ is implicitly defined by a nested optimization problem. A common solution is to differentiate the KKT system that implicitly defines z^*_θ w.r.t. θ [7]. Another crucial point is the selection of the stochastic predictor in DFL, which in the paper we choose to use diffusion models to represent P_θ.

Diffusion probabilistic model

To generate complex multi-modal and high-dimensional distributions, diffusion probabilistic models [112] are a promising approach. Diffusion models couple a fixed noising chain with a learned reverse denoising chain. Let $y_0 \in \mathbb{R}^d$ denote a sample from the real data distribution $q(y_0)$ and $\{\beta_t \in (0,1)\}_{t=1}^T$ denote the noise schedule. Define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$. The *forward process* q adds Gaussian noise at each step t to y_1 through y_T:

$$q(y_t \mid y_{t-1}) = \mathcal{N}(y_t; \sqrt{1-\beta_t}y_{t-1}, \beta_t I), \quad t = 1, \ldots, T,$$

which guarantees that $q(y_T \mid y_0)$ converges to a standard normal distribution as $T \to \infty$ with common schedules ($\bar{\alpha}_T \to 0$). Note that y_t can be equivalently sampled without iterating through intermediate time steps: $y_t = \sqrt{\bar{\alpha}_t}y_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$, where $\epsilon \sim N(0, I)$ is a Gaussian noise.

In the *reverse process* p, the diffusion model predicts the unknown added noise by

$$p_\theta(y_{t-1} \mid y_t) = \mathcal{N}(y_{t-1};\ \mu_\theta(y_t, t),\ \sigma_t^2 I), \tag{5.2}$$

whose mean $\mu_\theta(\cdot, t)$ is parameterized by a neural network predictor and variance is either fixed ($\sigma_t^2 = \beta_t$) or learned. The combination of p and q is equivalent to a

[1]We consider affine constraints in our main paper for simplicity. The extension from affine constraints to general convex constraints follows a similar derivation as in the linear case.

hierarchical variational auto-encoder [272], and thus can be optimized by using the evidence lower bound (ELBO) as the loss function [113].

Conditional Diffusion Model. Throughout this chapter, x denotes contextual features, and every transition probability is conditioned on x [266]. We use $P_\theta(\cdot|x)$ for the diffusion model's conditional distribution for generated data and $p_\theta(y_{t-1} \mid y_t, x)$ for its Markov transitions.

5.4 Stochastic Optimization and Reparameterization Estimator

Real-world decision problems often face significant uncertainty in their parameters. Optimizing with a stochastic predictor (e.g., diffusion model) yields better results than deterministic optimization, by explicitly modeling the uncertainty and optimizing the expected cost. Figure 5.1 illustrates a simple example: any deterministic solution ends up at an extreme decision with a higher expected cost, while the stochastic solution averages costs across likely outcomes and selects an interior decision with a lower expected cost.

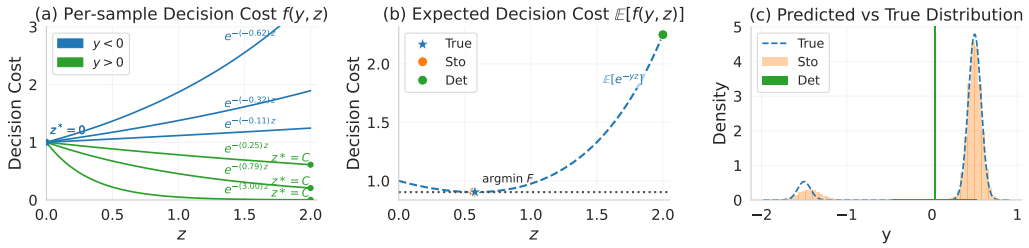


Figure 5.1: A comparison of deterministic vs. stochastic optimization with cost function $\exp(-yz)$, as described in Section 5.6. **(a)** Each curve represents a cost function given a sample y. For any fixed y, the deterministic optimization decision lies at one of the boundaries ($z^* = 0$ or $z^* = C$). **(b)** When averaging the cost function over many samples of y, the stochastic optimization decision lies in the interior of the feasible region instead of on the boundary. Thus, any deterministic optimization decision is suboptimal. **(c)** A probabilistic (diffusion) model captures a distribution over Y that closely resembles the true bimodal distribution.

Solving stochastic DFL. Formally, in the stochastic case, the optimality condition for the decision problem must consider an expectation. The stationarity condition for decision problem Eq. 5.1 becomes

$$\nabla_z \mathcal{L}(\theta, z^*, \lambda^*, \nu^*; x) = \mathbb{E}_{y \sim P_\theta(\cdot|x)}[\nabla_z f(y, z)] + G^\top \lambda^* + A^\top \nu^* = 0,$$

where $\mathcal{L}$ denotes the Lagrangian. Note that the dependency on θ in the stationarity condition is in the distribution. Therefore, we need to handle this dependency

carefully while differentiating the KKT system with respect to θ:

$$\underbrace{\frac{\partial}{\partial\theta}(\nabla_z \mathcal{L}(\theta, z^*, \lambda^*, \nu^*; x))}_{\text{distributional gradient}} = \frac{\partial}{\partial\theta}(\mathbb{E}_{y\sim P_\theta(\cdot|x)}[\nabla_z f(y,z)] + G^\top\lambda^* + A^\top\nu^*) = 0. \tag{5.3}$$

To resolve the dependence of both the predictive distribution $P_\theta(y \mid x)$ and the decision z^* on θ, we first adopt the *reparameterization trick* [132] for the diffusion model. From Section 5.3, recall that the diffusion sampling process introduces Gaussian noise at each step. Thus, we can reparameterize the reverse process by fixing all the random draws (Gaussian noises). Formally, a sample $y \sim P_\theta(y \mid x)$ can be expressed as a transformation $y = R(\epsilon, \theta \mid x)$ of a base Gaussian noise sample $\epsilon \sim P(\epsilon)$, where R is differentiable in θ. This makes the diffusion sampling a deterministic function of θ. Then we have

$$\nabla_\theta \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)] = \mathbb{E}_{\epsilon\sim P(\epsilon)}[(\nabla_\theta R(\epsilon,\theta|x))^\top \nabla_y f(y,z)]. \tag{5.4}$$

Next, we incorporate this into the optimization. Following Eq. 5.3, we can formalize a KKT system that contains derivatives of $z^*_\theta, \lambda^*, \nu^*$ and $\nabla_\theta \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)]$. Plugging the reparameterized gradient estimator into the KKT system, we can solve for $\frac{\mathrm{d}z^*_\theta}{\mathrm{d}\theta}$ and then obtain the total derivative of the final objective F by multiplying $\frac{dF}{dz^*_\theta}$ [73] (see proof in Section 5.9):

$$\frac{dF}{d\theta} = -\begin{bmatrix} \frac{dF}{dz^*_\theta} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & A^\top \\ D(\lambda^*)G & D(Gz^*_\theta - h) & 0 \\ A & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_{\epsilon\sim P(\epsilon)}[(\nabla_\theta R(\epsilon,\theta|x))^\top \nabla^2_{zy} f(y, z^*_\theta)] \\ 0 \\ 0 \end{bmatrix}, \tag{5.5}$$

where $H = \mathbb{E}_{y\sim P_\theta(\cdot|x)}[\nabla^2_{zz} f(y, z^*_\theta)]$ is the Hessian of the Lagrangian with respect to z, and $D(v)$ denotes a diagonal matrix with v on its diagonal. In practice, one can sample ϵ from a certain distribution (e.g., Gaussian) multiple times to estimate the expectation and then obtain the gradient. This gives us reparameterization-based diffusion DFL using Eq. 5.5 to run stochastic DFL optimization.

5.5 Score Function Estimator

A major obstacle to implementing the total gradient (Eq. 5.5) is the need to back-propagate through the diffusion sampling process. In most cases, the diffusion model's generative process is complex and multi-step (e.g., 1000 steps), which makes

backpropagating through all those steps memory-intensive and prone to instability. To address this, we propose a **score function**[2] gradient estimator for the diffusion model, which circumvents explicit backpropagation through all sampling steps. The key idea is to rewrite the Jacobian $\nabla_\theta y$ in terms of the score $\nabla_\theta \log P_\theta(y \mid x)$, and then approximate the score with the diffusion model's ELBO training loss.

Transform the Jacobian into Score Function

We begin by rewriting the gradient of expectation as an expectation of a score function using the *log-trick* [180]. Formally, if $y \sim P_\theta(\cdot|x)$ and f is a function not dependent on θ, then by the log-derivative trick we have

$$\nabla_\theta \, \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)] = \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z) \cdot \nabla_\theta \log P_\theta(y \mid x)]. \tag{5.6}$$

Intuitively, instead of differentiating the output y through each diffusion step, we only need to compute the gradient for the final log-likelihood, which avoids the need to differentiate through the diffusion sampling process and yields an efficient estimator for the gradient.

Then, one remaining difficulty is that directly computing the exact $\nabla_\theta \log P_\theta(y \mid x)$ is complicated in practice because $P_\theta(y \mid x)$ is defined as the marginal probability of y after integrating out the latent diffusion trajectory. To obtain a computationally efficient estimator, we use the diffusion model's training objective as a surrogate for the log-likelihood. Specifically, diffusion models are typically trained by maximizing an ELBO that lower-bounds the log-likelihood:

$$\begin{aligned}
\log P_\theta(y_0) &= \log \int p_\theta(y_0|y_1)\, p_\theta(y_1|y_2) \cdots p_\theta(y_{T-1}|y_T)\, p_\theta(y_T)\, dy_{1:T} \\
&= \log \mathbb{E}_{y_t\sim q(y_t|y_{t-1})\forall t\in[T]} \left[\prod_{t=1}^{T} \frac{p_\theta(y_{t-1}|y_t)}{q(y_t|y_{t-1})} p_\theta(y_T) \right] \\
&\geq \mathbb{E}_{y_t\sim q(y_t|y_{t-1})\forall t\in[T]} \left[\sum_{t=1}^{T} \log \frac{p_\theta(y_{t-1}|y_t)}{q(y_t|y_{t-1})} + \log p_\theta(y_T) \right] \qquad (5.7) \\
&=: \mathrm{ELBO}(y_0;\theta), \qquad (5.8)
\end{aligned}$$

where we have applied Jensen's inequality. To approximate $\nabla_\theta \log P_\theta(y_0 \mid x)$ conditioned on x, we use the gradient of the conditional ELBO loss as a surrogate:

$$\nabla_\theta \log P_\theta(y_0 \mid x) \approx \nabla_\theta \, \mathrm{ELBO}(y_0 \mid x;\theta). \tag{5.9}$$

[2] In this chapter, the score function refers to the statistical score $\nabla_\theta \log P_\theta(y \mid x)$ (gradient of log-likelihood w.r.t. model parameters), as opposed to Stein's score $\nabla_y p(y_t \mid y_{t-1}, x)$ often used in the diffusion literature.

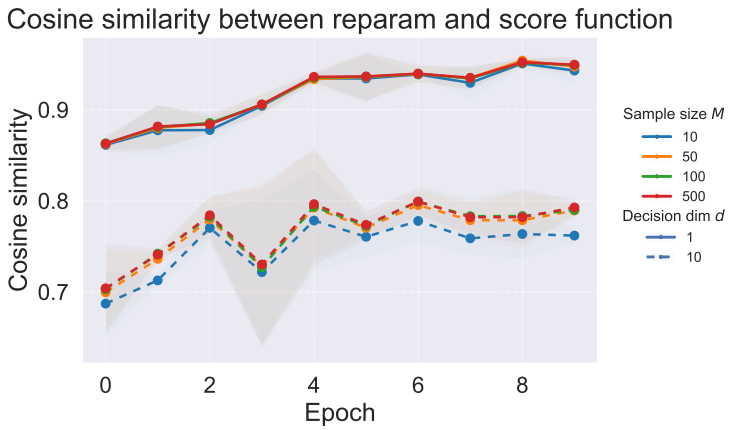


Figure 5.2: Cosine similarity between the reparameterization and score function gradient across different dimensions.

To motivate this approximation, we provide a theoretical justification for the surrogate ELBO gradient.

Proposition 5 (ELBO gradient approximation error)**.** *Denote the joint score function by* $s_\theta(z; y_0) \coloneqq \nabla_\theta \log p_\theta(y_0, z)$. *If* $\sup_z \|s_\theta(z; y_0)\| \le B(\theta, y_0)$ *for some finite upper bound* $B(\theta, y_0) > 0$*, then*

$$\begin{aligned}&\|\nabla_\theta \log P_\theta(y_0) - \nabla_\theta \,\mathrm{ELBO}(y_0; \theta)\| \\ &\le \sqrt{2} B(\theta, y_0) \sqrt{\mathrm{KL}(q(y_{1:T} \mid y_0) \parallel p_\theta(y_{1:T} \mid y_0))}.\end{aligned}$$

See Section 5.9 for proof and more details. Consequently, whenever the variational posterior (diffusion reverse process) is a good approximation of the target distribution (diffusion forward process), so that the KL divergence term is small, the ELBO gradient is guaranteed to be close to the true score. This justifies our use of $\nabla_\theta \,\mathrm{ELBO}(y_0; \theta)$ as a surrogate for $\nabla_\theta \log P_\theta(y_0)$.

In practice, we first sample a final output y from the diffusion model given contextual features x. We then sample a subset of k timesteps $\{t_1, t_2, \ldots, t_k\}$ ($k \ll T$) and run forward noising process q to generate the trajectory $\{y_{t_1}, y_{t_2}, \ldots, y_{t_k}\}$. As in DDPM [112], we adopt the simplified form of ELBO $\approx \mathbb{E}_{t\sim[T], y_0, \epsilon_t}[\|\epsilon_t - \epsilon_\theta(y_t, t)\|^2]$. We evaluate the ELBO on the sampled trajectories and compute its gradient w.r.t. θ as an estimation of the true score.

Empirical evidence suggests that the ELBO gradient closely tracks the true score, as shown in Figure 5.2, making Eq. 5.9 a reliable proxy in practice.

Overall Gradient for Score Function

By plugging the ELBO gradient approximation from Eq. 5.9 into Eq. 5.6, we can express the KKT conditions without using reparameterization and thus obtain the score function-based derivative:

$$\frac{dF}{d\theta} \approx -\begin{bmatrix} \frac{dF}{dz_\theta^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & A^\top \\ D(\lambda^*)G & D(Gz_\theta^* - h) & 0 \\ A & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_{y\sim P_\theta(\cdot|x)}[\nabla_z f(y, z_\theta^*)(\nabla_\theta \text{ELBO}(y|x;\theta))^\top] \\ 0 \\ 0 \end{bmatrix}. \tag{5.10}$$

Practical algorithm – weighted ELBO gradient. To compute the score surrogate in practice, we found it convenient to treat the total gradient as an importance-weighted form:

$$\frac{dF}{d\theta} \approx \frac{d}{d\theta}\,\mathbb{E}_{y\sim P_\theta(\cdot|x)}[\underbrace{\text{detach}[w_\theta(y)]}_{\text{importance weight, no grad in } \theta} \cdot \underbrace{\text{ELBO}(y|x,\theta)}_{\text{1-step forward}}], \tag{5.11}$$

where $w_\theta(y)$ is the importance weight simplified from Eq. 5.10 (see Section 5.9 for complete form). This yields a *weighted-ELBO* gradient estimator: we treat $w_\theta(y)$ as a stop-gradient weight and only differentiate the ELBO w.r.t. θ, greatly reducing computations. We implement the entire gradient computation as a user-friendly PyTorch autograd module: the forward pass returns the optimal decision z^* (and λ^*, ν^*), and the backward pass computes the gradient $\frac{dF}{d\theta}$ as derived above.

Variance-reduction strategy. While the score-function estimator is effective, a naive implementation of the weighted ELBO loss in Eq. 5.11 can suffer from high variance, leading to unstable training. In practice, we found that carefully designing the sampling strategy for the ELBO loss is crucial to obtaining low-variance and stable gradients. To reduce the variance, we utilize the method from Improved DDPM [188] for choosing diffusion steps. Specifically, instead of uniform sampling, we use *importance sampling* over timesteps with probability p_t and weights $1/p_t$:

$$\nabla_\theta \text{ELBO}^{\text{IS}} = \mathbb{E}_{t\sim p_t}\left[\frac{\nabla_\theta(\text{ELBO}_t)}{p_t}\right],$$
$$\text{where } p_t \propto \sqrt{\mathbb{E}[\|\nabla_\theta(\text{ELBO}_t)\|^2]} \text{ and } \sum_t p_t = 1.$$

This method remains unbiased, but the variance is minimized. In essence, this approach gives less weight to the early timesteps that have large gradients and more weight to later timesteps.

5.6 Experiments

We evaluate the performance of our diffusion-based DFL approaches on a variety of tasks, comparing against several baseline methods. Specifically, we consider:

- **Two-stage predict-then-optimize baselines**: a deterministic MLP, a Gaussian probabilistic model, and a diffusion model trained to minimize prediction error [78].
- **Deterministic DFL**: a deterministic MLP model with end-to-end DFL training [73].
- **Gaussian DFL** (both reparameterization and score function): a Gaussian probabilistic model with end-to-end stochastic DFL training; see details in Appendix 5.9.
- **Offline Contextual Bandits**: solving the full-information contextual bandit using policy-based learning [41]. Note that this method ignores the constraints of the optimization. See details in Appendix 5.9.
- **Diffusion DFL (ours)**: our diffusion model predictor, trained with either reparameterization or score-function gradient estimators.

Architectural and training details for all methods are summarized in Section 5.9, and code for our experiments is available on GitHub.[3]

Synthetic Product Allocation Task

In this example, we consider a factory that decides how much to manufacture for each of $d \in \mathbb{N}$ products. The parameter $Y \in \mathbb{R}^d$ represents the *profit margin* for each product, i.e., Y_i is the profit per unit of product i; due to uncertainty in market conditions, Y is uncertain. The factory's decision $z \in [0, C]^d$ represents how much of each product to manufacture, where C is the maximum capacity for each product. For simplicity, we do not consider any contextual features x in this example. That means DFL learns a distribution that generates y that can minimize the decision objective.

Suppose that the factory has a risk-averse cost function $f(y, z) = \exp(-y^\top z)$, which indicates that the factory wants to put a larger weight on the product with higher profit[4] Y_i. Under uncertainty, the decision-maker seeks to minimize the *expected cost* by solving a stochastic optimization problem:

$$z^*_{\text{sto}} \in \underset{z \in [0,C]^d}{\arg\min}\, \mathbb{E}_{y \sim P_\theta(\cdot|x)}[\exp(-y^\top z)].$$

[3] https://github.com/GT-KOALA/Diffusion_DFL

[4] Here, we have ignored the degenerate case $y = 0$. To deal with the degenerate case, one could add a zero-centered bump function $c(y)$ to the objective $f(y, z)$.

In this stochastic case, the optimal investment z^*_{sto} typically lies in the interior of the feasible region, which balances the potential high reward of investing against the risk of losses.

Experimental setup. We simulate the uncertain parameter Y drawn from a mixture of Gaussians,

$$Y_i \overset{\text{iid}}{\sim} p \cdot \mathcal{N}(a, \sigma^2) + (1-p) \cdot \mathcal{N}(-b, \sigma^2).$$

Specifically, we set $p = 0.8, a = 1, b = 3, \sigma = 0.15, C = 2$. We train each model (deterministic, Gaussian, diffusion) on this distribution in a decision-focused manner (for DFL methods) or on pure regression (for two-stage), and evaluate the expected cost achieved by the resulting decision $z^*_\theta(x)$. We present the results of one product ($d = 1$) in Figure 5.1 and 10 products ($d = 10$) in Table 5.1.

Power Scheduling Task

In this experiment, we evaluate our method on a real-world energy scheduling problem from Donti et al. [73]. This task involves a 24-hour generation-scheduling problem in which the operator chooses $z \in \mathbb{R}^{24}$ (hourly generation). Given a realization y of demand, the decision loss penalizes shortage and excess with asymmetric linear costs (γ_s and γ_e) plus a quadratic tracking term; the decision must also satisfy a ramping bound c_r. Let $[v]_+ := \max(v, 0)$. We have the decision loss as the quadratic problem:

$$\min_z \ \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)] = \sum_{i=1}^{24} \mathbb{E}_{y\sim P_\theta(\cdot|x)}\left[\gamma_s[y_i - z_i]_+ + \gamma_e[z_i - y_i]_+ + \frac{1}{2}(z_i - y_i)^2\right],$$
$$\text{s.t. } |z_i - z_{i-1}| \le c_r \text{ for all } i \in \{1, 2, \ldots, 24\}.$$

Experimental setup. We use more than 8 years of historical data from a regional power grid [204]. Feature x includes the previous day's hourly load, temperature, next-day temperature forecasts, non-linear transforms (lags and rolling statistics), calendar indicators, and yearly sinusoidal features. Given x, the prediction model $P_\theta(\cdot|x)$ outputs a distribution over $y \in \mathbb{R}^{24}$. We report the test decision cost in Table 5.1 and a held-out horizon in Figure 5.7.

Stock Market Portfolio Task

In this experiment, we apply our diffusion DFL approach to a financial portfolio optimization problem under uncertain stock returns. Here, the random vector $y \in \mathbb{R}^n$ represents the returns for the n assets on the next day, and the decision $z \in \mathbb{R}^n$

represents the portfolio weights allocated to those assets. We consider a mean-variance trade-off decision loss: maximize expected return while keeping the risk (variance) low. This can be written as minimizing a loss that is a negative expected return plus a quadratic penalty on variance:

$$\min_{z} \ \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)] = \mathbb{E}_{y\sim P_\theta(\cdot|x)}\left[\frac{\alpha}{2} z^\top y y^\top z - y^\top z\right]$$
$$\text{s.t. } z^\top \mathbf{1} = 1,\ 0 \le z_i \le 1,$$

where $\alpha > 0$ is a risk parameter and constraints enforce that z is a valid portfolio. In practice, the deterministic solution may concentrate heavily on a few assets and yield a low average return, whereas a stochastic approach can achieve higher returns by accounting for variance.

Experimental setup. We have daily prices and volumes spanning 2004-2017 and evaluate on the S&P 500 index constituents [212]. The features $x \in \mathbb{R}^{28}$ include recent historical return, trading volume windows, and rolling averages. The immediate-return predictor $P_\theta(\cdot|x)$ predicts the next day's price. We report the performance of different DFL baselines with 50 portfolios in Table 5.1 and other sizes of portfolios in Section 5.7.

Table 5.1: Results for different optimization tasks. Our two diffusion DFL methods achieve the best and second-best decision quality in all 3 tasks, significantly better than other baselines. **Bolded** values are the best in test task losses; underlined values are the 2nd-best. Mean ± standard error across 10 runs.

	Synthetic Example		Power Schedule		Stock Portfolio	
Label / Method	**RMSE↓**	**Task↓**	**RMSE↓**	**Task↓**	**RMSE↓**	**Task (%)↑**
Two-stage (TS)						
Deterministic TS	$0.639_{\pm 0.00}$	$1.987_{\pm 0.00}$	$0.120_{\pm 0.00}$	$41.239_{\pm 3.18}$	$0.027_{\pm 0.00}$	$0.04\%_{\pm 0.04}$
Gaussian TS	$0.720_{\pm 0.00}$	$1.272_{\pm 0.23}$	$0.117_{\pm 0.00}$	$5.580_{\pm 0.45}$	$0.188_{\pm 0.03}$	$0.10\%_{\pm 0.04}$
Diffusion TS	$0.905_{\pm 0.00}$	$0.393_{\pm 0.00}$	$0.147_{\pm 0.00}$	$7.901_{\pm 0.76}$	$0.455_{\pm 0.00}$	$0.13\%_{\pm 0.03}$
Offline Contextual Bandits						
Policy-based Learning	$0.873_{\pm 0.00}$	$0.568_{\pm 0.02}$	$4.712_{\pm 0.10}$	$4.440_{\pm 0.17}$	$0.064_{\pm 0.00}$	$1.74\%_{\pm 0.41}$
Decision-focused learning (DFL)						
Deterministic	$0.640_{\pm 0.00}$	$1.987_{\pm 0.00}$	$4.997_{\pm 0.10}$	$4.324_{\pm 0.25}$	$0.032_{\pm 0.00}$	$0.07\%_{\pm 0.00}$
Gaussian Reparameterization	$0.707_{\pm 0.00}$	$1.169_{\pm 0.03}$	$4.525_{\pm 0.12}$	$3.724_{\pm 0.05}$	$0.189_{\pm 0.03}$	$0.08\%_{\pm 0.03}$
Gaussian Score Function	$0.708_{\pm 0.00}$	$1.132_{\pm 0.00}$	$4.713_{\pm 0.15}$	$4.087_{\pm 0.06}$	$0.187_{\pm 0.03}$	$0.14\%_{\pm 0.05}$
Diffusion Reparameterization	$0.852_{\pm 0.01}$	$\underline{0.365}_{\pm 0.00}$	$3.141_{\pm 0.06}$	$\mathbf{3.152}_{\pm 0.03}$	$0.063_{\pm 0.00}$	$\mathbf{4.17\%}_{\pm 0.24}$
Diffusion Score Function	$0.849_{\pm 0.09}$	$\mathbf{0.362}_{\pm 0.00}$	$2.893_{\pm 0.03}$	$\underline{3.171}_{\pm 0.02}$	$0.067_{\pm 0.00}$	$\underline{3.98\%}_{\pm 0.31}$

5.7 Discussion of Experimental Results and Ablation Study

Discussion of Results in Table 5.1

Two-stage vs. DFL. As shown in Table 5.1, across all three experiment tasks, we find that end-to-end DFL leads to better downstream decisions than the conventional two-stage approach. Conventional two-stage methods minimize RMSE during training, but this often leads to poor downstream decisions. In contrast, all variants of DFL directly minimize the decision cost during training and thus achieve lower decision costs.

Offline Contextual Bandits vs. DFL. Table 5.1 shows that the policy-based offline bandit method can outperform two-stage approaches on tasks such as power scheduling and portfolio tasks, since it directly optimizes the downstream task objective rather than training a predictor with a surrogate loss. It is also competitive with deterministic and Gaussian DFL variants. One reason is that, in some settings (such as the synthetic task), DFL with deterministic optimization (the Gaussian case can be written in closed-form as a deterministic optimization) fails to represent multi-modal distributions on uncertain parameters and is therefore a fundamentally poor policy class that does not include the optimal policy. An expressive enough OCB policy class, on the other hand, can learn a strong context-to-action policy.

Nevertheless, Diffusion-DFL remains consistently superior to offline contextual bandits: by incorporating the known optimization structure and coupling a flexible generative model with the downstream solver, it can better capture complex decision distributions and deliver higher-quality decisions.

Deterministic vs. Stochastic Optimization. Our results show that stochastic DFL methods outperform deterministic DFL in terms of decision quality on every task. By modeling uncertainty, stochastic predictors enable the decision optimization to account for risk and variability in outcomes. For instance, in the portfolio experiment, the deterministic DFL yields only 0.07% return, whereas a Gaussian DFL modestly improves that, and our diffusion DFL achieves nearly 4% average return. These gains come from the stochastic models' ability to predict uncertainty: instead of committing to a point prediction of y, the stochastic DFL produces decisions for a range of possible outcomes.

Benefits of Diffusion DFL. Among the stochastic approaches, including baselines using Gaussian models, our diffusion DFL method consistently delivers the best

decision performance. In particular, the diffusion model's strength is the capacity to capture complex, multi-modal outcome distributions that a simple parametric Gaussian cannot represent. The Gaussian DFL sometimes falls short of the optimal decision quality. The diffusion model, on the other hand, can represent more intricate distributions of y, leading to decisions that better reflect complex scenarios.

Ablation Study

Comparison Cost for Reparameterization and Score function. A key finding from our ablation study is the computational advantage of the score-function approach over the reparameterization. Here, we measure the trade-off between training cost and the final decision performance for different gradient estimators and sampling budgets.

In Figure 5.3 (a), we see that all variants reach similar final performance on the test set, indicating that even using as few as 50 samples is sufficient to optimize the decision quality accurately. Figure 5.4 plots the GPU memory cost alongside the final test loss. The reparameterization method is very computationally expensive, requiring about 60 GB of GPU memory for backpropagating through all diffusion steps. In contrast, the score-function with 50 samples achieves virtually the same test loss as the reparameterization method (difference within 0.02) while using an order of magnitude less memory. Even with 10 samples, though slightly worse in loss, it still outperforms the deterministic baseline and uses a tiny fraction of the compute. These results validate that the score-function approach retains the decision-quality benefits of diffusion DFL while dramatically cutting computational requirements, making diffusion DFL practical even for complex problems.

Gradient variance reduction. As discussed in Section 5.5, using the score function estimator allows us to avoid backpropagating through the entire diffusion sampling process by only sampling a limited number of diffusion timesteps per update. The reason behind this is that a naive implementation, sampling timesteps uniformly at random, would yield a very high variance in the gradient estimates, which then leads to unstable training. Intuitively, early diffusion steps (large noise levels) dominate the ELBO loss and its gradients, so if they happen to be sampled, they contribute disproportionately and noisily. With a small random subset of timesteps, the gradient estimate can thus be highly imbalanced and noisy, which causes training divergence in practice.

To address this, we adopt an importance sampling strategy for choosing diffusion

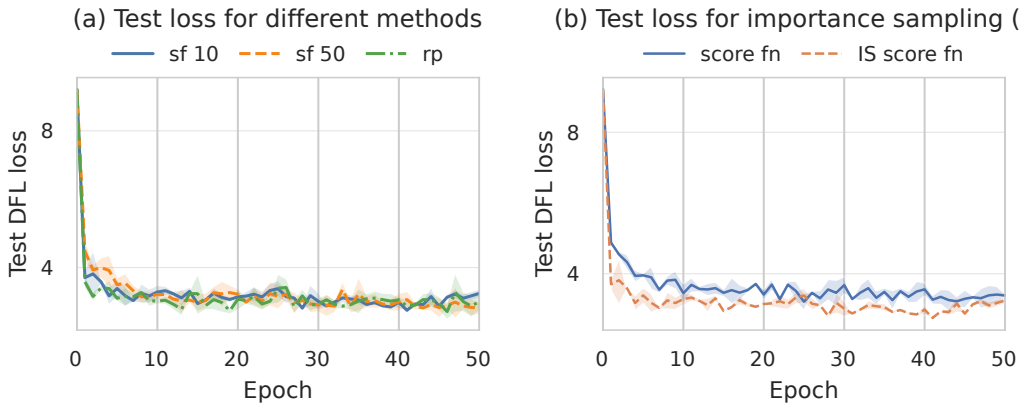


Figure 5.3: Learning curves for (a) score function with 10 and 50 samples (*sf 10* and *sf 50*) and reparameterization (*rp*), (b) score function and importance-weighted score function with 10 samples.

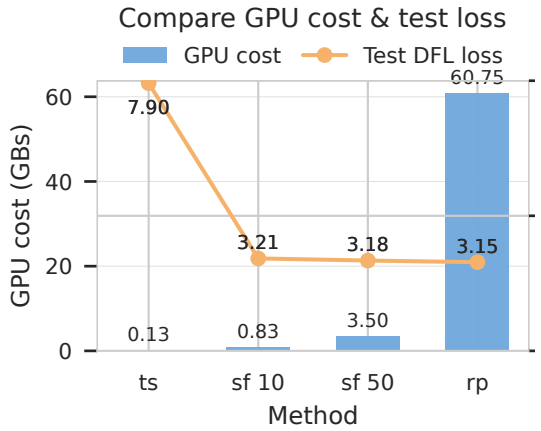


Figure 5.4: Computation cost vs. performance trade-off for diffusion DFL training

timesteps. Empirically, as shown in Figure 5.3 (b), the learning curves with the importance-weighted sampler are much smoother and more stable than with the uniform sampler. The score-function DFL training no longer diverges; instead, it converges cleanly, indicating that our variance reduction strategy successfully stabilizes the training process for diffusion DFL.

Comparison on different problem sizes. A key challenge for DFL is scalability: as the decision dimension grows, many methods degrade significantly [175]. In this experiment, we investigate the performance of DFL methods under various decision dimensions in the stock portfolio. Specifically, we set the decision dimension range from 10 to 100 and report the test regrets. As summarized in Figure 5.5, the regret gap between diffusion-DFL and Gaussian and deterministic methods increases with the

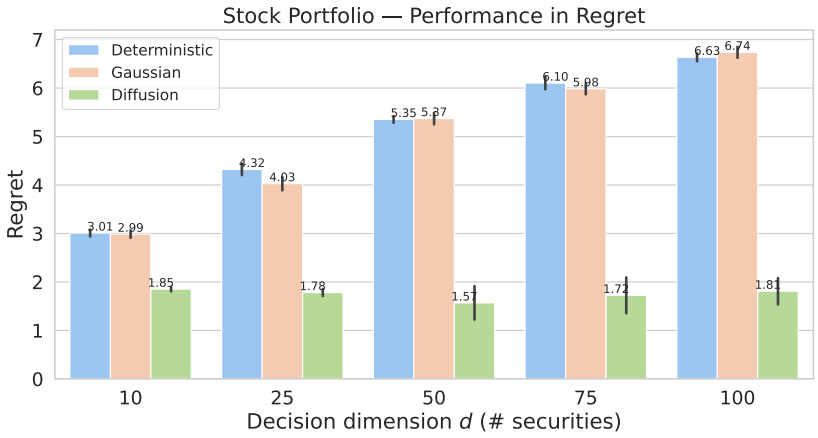


Figure 5.5: Test regret vs. decision dimension d in the stock portfolio task.

growth of dimensionality, which demonstrates that diffusion DFL scales effectively in more complex decision settings.

5.8 Conclusion

We propose the first diffusion-based DFL approach for stochastic optimization, which trains a diffusion model to capture complex uncertainty in problem parameters. We develop two end-to-end training techniques to integrate the diffusion model into decision-making: reparameterization and score function approximation. As demonstrated with empirical evidence, the score function method drastically reduces memory and computation cost while achieving similar performance to reparameterization and being easy to train. Empirically, diffusion DFL achieves state-of-the-art results on multiple benchmarks, consistently outperforming both traditional two-stage methods and prior DFL approaches.

5.9 Appendix

Notation. We let $\frac{\partial f}{\partial x}$ denote the Jacobian matrix where $\left(\frac{\partial f}{\partial x}\right)_{i,j} := \frac{\partial f_i}{\partial x_j}$ and $\nabla_x f := \left(\frac{\partial f}{\partial x}\right)^\top$ denote the gradient. For a vector v, $D(v)$ denotes a diagonal matrix with v on its diagonal. Let $P(\cdot)$ denote a probability distribution and $p(\cdot)$ denote a probability density; in particular, for diffusion models we use P_θ for the model's output distribution and p_θ for transition densities.

In this appendix, we derive the decision optimization problem with **general convex constraints** rather than merely linear constraints. Assume the optimization problem

is

$$z_\theta^*(x) = \arg\min_z \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)], \quad \text{s.t. } h(x,z) \le 0,\ g(x,z) = 0,$$

where $h(x,z) \le 0$ denotes the convex inequalities constraints and $g(x,z) = 0$ denotes the equality constraints.

Proofs for Section 5.4

Proposition 6 (Reparameterization trick in diffusion models)**.** *Let $T \in \mathbb{N}^+$, and suppose the reverse diffusion model defines a Gaussian distribution in Eq. 5.2 with fixed scalars $\sigma_t \ge 0$ and a standard normal prior $y_T \sim \mathcal{N}(0, I)$. Let $\{\epsilon_t\}_{t=0}^T$ be i.i.d. $\mathcal{N}(0, I)$. Then the model output y can be expressed as a transformation $y = R(\epsilon_{0:T}, \theta \mid x)$ of a base noise distribution $\epsilon \sim P(\epsilon)$, where R is differentiable in θ. Also assume $\mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)]$ is continuously differentiable. Then we have*

$$\nabla_\theta \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)] = \mathbb{E}_{\epsilon\sim P(\epsilon)}\left[\left(\sum_{s=1}^{T}\left(\prod_{u=1}^{s-1} J_u\right)A_s\right)^\top \nabla_y f(R(\epsilon,\theta\mid x), z)\right],$$

where $A_t := \frac{\partial \mu_\theta(y_t,t,x)}{\partial \theta}$, $J_t := \frac{\partial \mu_\theta(y_t,t,x)}{\partial y_t}$, and we define $\prod_{u=1}^{0} J_u := I$.

Proof. The conditional diffusion reverse process is defined as

$$y_{t-1} = \mu_\theta(y_t, t, x) + \sigma_t \epsilon_{t-1}, \qquad y_T = \epsilon_T,$$

where the noise term $\sigma_t\epsilon_{t-1}$ is θ-independent. Differentiating both sides w.r.t. θ gives

$$\frac{\partial y_{t-1}}{\partial \theta} = \frac{\partial \mu_\theta(y_t,t,x)}{\partial \theta} + \frac{\partial \mu_\theta(y_t,t,x)}{\partial y_t}\frac{\partial y_t}{\partial \theta}.$$

Denote

$$A_t := \frac{\partial \mu_\theta(y_t,t,x)}{\partial \theta}, \quad J_t := \frac{\partial \mu_\theta(y_t,t,x)}{\partial y_t}, \quad G_t := \frac{\partial y_t}{\partial \theta}.$$

Thus, we have

$$G_{t-1} = A_t + J_t G_t, \quad G_T = 0.$$

Our final goal is:

$$\begin{aligned}\nabla_\theta R(\epsilon_{0:T}, \theta|x) &= \frac{\partial y_0}{\partial \theta} = G_0 \\ &= A_1 + J_1A_2 + J_1J_2A_3 + \cdots + J_1\cdots J_{T_1}A_t \\ &= \sum_{s=1}^{T}\left(\prod_{u=1}^{s-1} J_u\right)A_s,\end{aligned}$$

where we define $\prod_{u=1}^{0} J_u := I$. Then, we have

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)] &= \nabla_\theta \, \mathbb{E}_{\epsilon\sim P(\epsilon)}[f(R(\epsilon,\theta|x),z)] \\
&= \mathbb{E}_{\epsilon\sim P(\epsilon)}[\nabla_\theta f(R(\epsilon,\theta|x),z)] \\
&= \mathbb{E}_{\epsilon\sim P(\epsilon)}\left[\nabla_\theta R(\epsilon,\theta \mid x)^\top \nabla_y f(R(\epsilon,\theta|x),z)\right] \\
&= \mathbb{E}_{\epsilon\sim P(\epsilon)}\left[\left(\sum_{s=1}^{T}\left(\prod_{u=1}^{s-1} J_u\right) A_s\right)^\top \nabla_y f(R(\epsilon,\theta|x),z)\right]
\end{aligned}$$

□

Lemma 9 (Gradient of Reparameterization method)**.** *Assume the model prediction y can be expressed as a transformation $y = T(\epsilon,\theta \mid x)$, $\epsilon \sim P(\epsilon)$. The total derivative of the decision objective F w.r.t. θ can be computed as*

$$\frac{dF}{d\theta} = -\begin{bmatrix} \frac{dF}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x,z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_{\epsilon\sim P(\epsilon)}[(\nabla_\theta T(\epsilon,\theta|x))^\top \nabla^2_{zy} f(z^*,y)] \\ 0 \\ 0 \end{bmatrix},$$

where $H = \mathbb{E}_{y\sim P_\theta(\cdot|x)}[\nabla^2_{zz} f(y,z^*)] + \nabla^2_{zz}(\lambda^{*\top} h(x,z^*))$ is the Hessian of the Lagrangian with respect to z, $G = \nabla_z h(x,z^*)$ is the Jacobian of the inequality constraints in z^*, and $Q = \nabla_z g(x,z^*)$ is the Jacobian of the equality constraints in z^*.

Proof. At the primal-dual optimal solution $(z^*_\theta, \lambda^*_\theta, \nu^*_\theta)$ to Eq. 5.1, the following KKT conditions must hold:

$$\begin{aligned}
\nabla_z \mathcal{L}(\theta, z^*_\theta, \lambda_\theta, \nu_\theta; x) &= 0, \\
\lambda_\theta \odot h(x, z^*_\theta) &= 0, \\
g(x, z^*_\theta) &= 0 \\
\lambda_\theta \geq 0, \nu_\theta &\geq 0, \\
h(x, z^*_\theta) &\leq 0.
\end{aligned}$$

Since h does not depend on θ here, we can apply Proposition 6 to the KKT conditions

to get

$$\begin{aligned}
&\frac{\partial \nabla_z \mathcal{L}}{\partial \theta} + \frac{\partial \nabla_z \mathcal{L}}{\partial z}\frac{\partial z^*}{\partial \theta} + \frac{\partial \nabla_z \mathcal{L}}{\partial \lambda^*}\frac{\partial \lambda^*}{\partial \theta} + \frac{\partial \nabla_z \mathcal{L}}{\partial \nu^*}\frac{\partial \nu^*}{\partial \theta} \\
&= \mathbb{E}_{\epsilon \sim P(\epsilon)}[(\nabla_\theta T(\epsilon, \theta|x))^\top \nabla_y (\nabla_z f(z^*, y))] \\
&\quad + \left(\mathbb{E}_{y \sim P_\theta(\cdot|x)}[\nabla^2_{zz} f(z^*, y)] + \nabla^2_{zz} h(x, z^*)\right)\frac{\partial z^*}{\partial \theta} \\
&\quad + \nabla_z h(x, z^*)\frac{\partial \lambda^*}{\partial \theta} + \nabla_z g(x, z^*)\frac{\partial \lambda^*}{\partial \theta} \\
&= 0.
\end{aligned}$$

$$\begin{aligned}
&\frac{\partial \lambda^* \odot h(x, z^*)}{\partial z^*}\frac{\partial z^*}{\partial \theta} + \frac{\partial \lambda^* \odot h(x, z^*)}{\partial \lambda^*}\frac{\partial \lambda^*}{\partial \theta} \\
&= D(\lambda^*)\nabla_z h(x, z^*)\frac{\partial z^*}{\partial \theta} + D(h(x, z^*))\frac{\partial \lambda^*}{\partial \theta} = 0.
\end{aligned}$$

In matrix form, we have

$$\begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x, z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial z^*}{\partial \theta} \\ \frac{\partial \lambda^*}{\partial \theta} \\ \frac{\partial \nu^*}{\partial \theta} \end{bmatrix} = - \begin{bmatrix} \mathbb{E}_{\epsilon \sim P(\epsilon)}[(\nabla_\theta T(\epsilon, \theta|x))^\top \nabla^2_{zy} f(z^*, y)] \\ 0 \\ 0 \end{bmatrix},$$

where $H = \mathbb{E}_{y \sim P_\theta(\cdot|x)}[\nabla^2_{zz} f(z^*, y)] + \nabla^2_{zz}(\lambda^{*\top} h(x, z^*)) + \nabla^2_{zz}(\nu^{*\top} g(x, z^*))$, $G = \nabla_z h(x, z^*)$, and $Q = \nabla_z g(x, z^*)$. Furthermore, if equalities and inequalities are affine (as in main paper), H reduces to $\mathbb{E}_{y \sim P_\theta(\cdot|x)}[\nabla^2_{zz} f(y, z^*)]$ since $\nabla^2_{zz} h = \nabla^2_{zz} g = 0$.

By chain rule, we have

$$\begin{aligned}
\frac{dF}{d\theta} &= \begin{bmatrix} \frac{dF}{dz^*} \\ 0 \end{bmatrix}^\top \begin{bmatrix} \frac{\partial z^*}{\partial \theta} \\ \frac{\partial \lambda^*}{\partial \theta} \end{bmatrix} \\
&= - \begin{bmatrix} \frac{dF}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x, z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_{\epsilon \sim P(\epsilon)}[(\nabla_\theta T(\epsilon, \theta|x))^\top \nabla^2_{zy} f(z^*, y)] \\ 0 \\ 0 \end{bmatrix}.
\end{aligned}$$

□

Proofs for Section 5.5

Proposition 7. *Let $P_\theta(y \mid x)$ be a probability density parameterized by $\theta \in \Theta$, and let $f : \mathcal{Y} \times \mathbb{R}^d \to \mathbb{R}$ be a scalar-valued function that does not depend on θ. Fix any $z \in \mathbb{R}^d$. Suppose that there exists some neighborhood $N(\theta_0) \subseteq \Theta$ around $\theta_0 \in \Theta$ such that the following 3 assumptions are satisfied:*

1. *For all $\theta \in N(\theta_0)$, the function $h(y) := P_\theta(y \mid x)\, f(y, z)$ is integrable;*
2. *For all $\theta \in N(\theta_0)$ and almost all $y \in \mathcal{Y}$, the gradient $\nabla_\theta P_\theta(y \mid x)$ exists; and*
3. *There exists an integrable function $g : \mathcal{Y} \to \mathbb{R}$ that dominates $\nabla_\theta P_\theta(y \mid x)$. That is, for all $\theta \in N(\theta_0)$ and almost all $y \in \mathcal{Y}$, $\|\nabla_\theta P_\theta(y \mid x)\|_1 \leq |g(y)|$.*

Then,

$$\nabla_\theta \mathbb{E}_{y \sim P_\theta(\cdot|x)}[f(y, z)] = \mathbb{E}_{y \sim P_{\theta_0}(\cdot|x)}[f(y, z) \cdot \nabla_\theta \log P_{\theta_0}(y \mid x)].$$

Proof. We make use of the log-derivative trick:

$$P_{\theta_0}(y \mid x) \cdot \nabla_\theta \log P_{\theta_0}(y \mid x) = \frac{P_{\theta_0}(y \mid x)}{P_{\theta_0}(y \mid x)} \cdot \nabla_\theta P_{\theta_0}(y \mid x) = \nabla_\theta P_{\theta_0}(y \mid x).$$

Then

$$\begin{aligned}
&\nabla_\theta\, \mathbb{E}_{y \sim P_{\theta_0}(\cdot|x)}[f(y, z)] \\
&= \nabla_\theta \int_{\mathcal{Y}} f(y, z) P_{\theta_0}(y \mid x)\, \mathrm{d}y \\
&= \int_{\mathcal{Y}} \nabla_\theta \left[f(y, z)\, P_{\theta_0}(y \mid x) \right] \mathrm{d}y && \text{(by Leibniz integral rule)} \\
&= \int_{\mathcal{Y}} f(y, z)\, P_{\theta_0}(y \mid x)\, \nabla_\theta \log P_{\theta_0}(y \mid x)\, \mathrm{d}y && \text{(by log-derivative trick)} \\
&= \mathbb{E}_{y \sim P_{\theta_0}(\cdot|x)} \left[f(y, z) \nabla_\theta \log P_{\theta_0}(y \mid x) \right].
\end{aligned}$$

□

Lemma 10 (Gradient of Score Function)**.** *The total derivative of the decision objective F w.r.t. θ can be computed as*

$$\frac{dF}{d\theta} = -\begin{bmatrix} \frac{dF}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x, z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_{y \sim P_\theta(\cdot|x)}[\nabla_z f(z^*, y)(\frac{dELBO}{d\theta})^\top] \\ 0 \\ 0 \end{bmatrix}.$$

Proof. Differentiating this KKT system w.r.t. θ and applying Proposition 7 yields

$$\begin{aligned}
&\frac{\partial \nabla_z \mathcal{L}}{\partial \theta} + \frac{\partial \nabla_z \mathcal{L}}{\partial z}\frac{\partial z^*}{\partial \theta} + \frac{\partial \nabla_z \mathcal{L}}{\partial \lambda^*}\frac{\partial \lambda^*}{\partial \theta} + \frac{\partial \nabla_z \mathcal{L}}{\partial \nu^*}\frac{\partial \nu^*}{\partial \theta} \\
&= \mathbb{E}_{y \sim P_\theta(\cdot|x)}[\nabla_z f(z^*, y)(\nabla_\theta \log P_\theta(y|x))^\top] \\
&\quad + (\mathbb{E}_{y \sim P_\theta(\cdot|x)}[\nabla^2_{zz} f(z^*, y)] + \nabla^2_{zz}(\lambda^{\star\top} h(x, z^*)))\frac{\partial z^*}{\partial \theta} \\
&\quad + \nabla_z h(x, z^*)\frac{\partial \lambda^*}{\partial \theta} + \nabla_z g(x, z^*)\frac{\partial \nu^*}{\partial \theta} = 0.
\end{aligned}$$

$$\frac{\partial \lambda^* \odot h(x, z^*)}{\partial z^*} \frac{\partial z^*}{\partial \theta} + \frac{\partial \lambda^* \odot h(x, z^*)}{\partial \lambda^*} \frac{\partial \lambda^*}{\partial \theta}$$
$$= D(\lambda^*) \nabla_z h(x, z^*) \frac{\partial z^*}{\partial \theta} + D(h(x, z^*)) \frac{\partial \lambda^*}{\partial \theta} = 0.$$

In matrix form, this becomes

$$\begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x, z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial z^*}{\partial \theta} \\ \frac{\partial \lambda^*}{\partial \theta} \\ \frac{\partial \nu^*}{\partial \theta} \end{bmatrix} = - \begin{bmatrix} \mathbb{E}_y[\nabla_z f(z^*, y)(\nabla_\theta \log P_\theta(y \mid x))^\top] \\ 0 \\ 0 \end{bmatrix}, \tag{5.12}$$

where $H = \mathbb{E}_{y \sim P_\theta(\cdot|x)}[\nabla^2_{zz} f(z^*, y)] + \nabla^2_{zz}(\lambda^{*\top} h(x, z^*))$, $G = \nabla_z h(x, z^*)$.

Applying the chain rule to F now gives

$$\frac{dF}{d\theta} = \begin{bmatrix} \frac{dF}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} \frac{\partial z^*}{\partial \theta} \\ \frac{\partial \lambda^*}{\partial \theta} \\ \frac{\partial \nu^*}{\partial \theta} \end{bmatrix}$$
$$= - \begin{bmatrix} \frac{dF}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x, z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_{y \sim P_\theta(\cdot|x)}[\nabla_z f(z^*, y)(\nabla_\theta \log P_\theta(y \mid x))^\top] \\ 0 \\ 0 \end{bmatrix}.$$

Then, we replace $\nabla_\theta \log P_\theta(y \mid x)$ with the gradient of ELBO score for sample y and have

$$\frac{dF}{d\theta} = - \begin{bmatrix} \frac{dF}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x, z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_{y \sim P_\theta(\cdot|x)}[\nabla_z f(z^*, y) \cdot (\nabla_\theta \mathrm{ELBO}(\theta; y))^\top] \\ 0 \\ 0 \end{bmatrix}.$$

□

Remark 1 (Why we cannot compute the gradient using score-matching)**.** *One may attempt to apply the chain rule* $\nabla_\theta \log P_\theta(y|x) = \nabla_\theta y \nabla_y \log P_\theta(y|x)$*, and then estimate* $\nabla_y \log P_\theta(y_{t+1}|x) \approx \nabla_y \log P_\theta(y_{t+1}|y_t, x) \approx s_\theta(y_t, t, x)$ *via score-matching [253] (using the learned score* $s_\theta(y_t, t, x)$ *of the diffusion model). However, this approach is invalid in our setting: Under the log-trick,* y *is treated as a free variable and* θ *enters only through* $P_\theta(y|x)$*, so the pathwise term* $\nabla_\theta y$ *does not exist (see Section 5.9 for derivation). Our ELBO-based surrogate (Eq. (5.9)) avoids this obstacle entirely.*

Proof for Eq. 5.11

Based on the results in Lemma 10, we have

$$\begin{aligned}
\frac{dF}{d\theta} &= -\begin{bmatrix} \frac{dF}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x,z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_{y\sim P_\theta(\cdot|x)}[\nabla_z f(z^*,y)\cdot(\nabla_\theta \text{ELBO}(\theta;y))^\top] \\ 0 \\ 0 \end{bmatrix} \\
&= \underbrace{-\begin{bmatrix} \frac{dF}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x,z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix}^{-1}}_{:=u(\theta)^\top} \frac{d}{d\theta} \begin{bmatrix} \mathbb{E}_{y\sim P_\theta(\cdot|x)}[\nabla_z f(z^*,y)\,\text{ELBO}] \\ 0 \\ 0 \end{bmatrix} \\
&= \frac{d}{d\theta}\mathbb{E}_{y\sim P_\theta(\cdot|x)}[\underbrace{u(\theta)^\top \begin{bmatrix} [\nabla_z f(z^*,y)] \\ 0 \\ 0 \end{bmatrix}}_{:=w_\theta(y)} \text{ELBO}] \\
&= \frac{d}{d\theta}\mathbb{E}_{y\sim P_\theta(\cdot|x)}[\text{detach}[w_\theta(y)]\text{ELBO}].
\end{aligned}$$

Gradient Error of ELBO vs. Log-likelihood

Under mild assumptions, we can prove an upper bound for the error of our ELBO gradient approximation. Recall from Eq. 5.7, we define ELBO as a lower bound for log-likelihood. Here, we can actually write an equality relationship between them:

$$\log P_\theta(y_0) = \text{ELBO}(y_0;\theta) + \text{KL}(q(z) \parallel p_\theta(z|y_0)),$$

where z denotes a latent variable and

$$\text{ELBO}(y_0,\theta) := \mathbb{E}_{q(z)}[\log p_\theta(x,z)] - \mathbb{E}_{q(z)}[\log q(z)].$$

Let denote the score function as $s_\theta(z;y_0) := \nabla_\theta \log p_\theta(y_0,z)$. We immediately have the following two results:

$$\begin{aligned}
\nabla_\theta \log P_\theta(y_0) &= \nabla_\theta \log \int p_\theta(y_0,z)\,dz \\
&= \frac{1}{p_\theta(y_0)}\int \nabla_\theta p_\theta(y_0,z)\,dz \\
&= \frac{1}{p_\theta(y_0)}\int p_\theta(y_0,z)\nabla_\theta \log p_\theta(y_0,z)\,dz \\
&= \int \nabla_\theta \log p_\theta(y_0,z)\frac{p_\theta(y_0,z)}{p_\theta(y_0)}\,dz \\
&= \mathbb{E}_{p_\theta(z|y_0)}[\nabla_\theta \log p_\theta(y_0,z)] \\
&= \mathbb{E}_{p_\theta(z|y_0)}[s_\theta(z;y_0)].
\end{aligned}$$

Similarly,

$$\begin{aligned}\nabla_\theta \,\mathrm{ELBO}(y_0;\theta) &= \nabla_\theta\, \mathbb{E}_{q(z)}[\log p_\theta(y_0, z)] \\ &= \mathbb{E}_{q(z)}[\nabla_\theta \log p_\theta(y_0, z)] \\ &= \mathbb{E}_{q(z)}[s_\theta(z; y_0)].\end{aligned}$$

Then we are ready to state the following proposition:

Proposition 8 (ELBO gradient approximation error)**.** *Denote the joint score function by* $s_\theta(z; y_0) \coloneqq \nabla_\theta \log p_\theta(y_0, z)$*. If* $\sup_w \|s_\theta(z; y_0)\| \le B(\theta, y_0)$ *for some finite upper bound* $B(\theta, y_0) > 0$*, then*

$$\begin{aligned}&\|\nabla_\theta \log P_\theta(y_0) - \nabla_\theta \,\mathrm{ELBO}(y_0;\theta)\| \\ &\le \sqrt{2} B(\theta, y_0) \sqrt{\mathrm{KL}(q(y_{1:T} \mid y_0) \parallel p_\theta(y_{1:T} \mid y_0))}.\end{aligned}$$

Proof.

$$\begin{aligned}\|\nabla_\theta \log P_\theta(y_0) - \nabla_\theta \,\mathrm{ELBO}(y_0;\theta)\| &= \left\|\mathbb{E}_{p_\theta(z|y_0)}[s_\theta(z; y_0)] - \mathbb{E}_{q(z)}[s_\theta(z; y_0)]\right\| \\ &= \left\|\int s_\theta(z; y_0)(p_\theta(z \mid y_0) - q(z))\, dz\right\| \\ &\le \int \|s_\theta(z; y_0)(p_\theta(z \mid y_0) - q(z))\|\, dz \\ &= \int \|s_\theta(z; y_0)\|\, |p_\theta(z \mid y_0) - q(z)|\, dz \\ &\le B(\theta, y_0) \int |p_\theta(z \mid y_0) - q(z)|\, dz \\ &= B(\theta, y_0) \cdot 2\mathrm{TV}(q(z), p_\theta(z \mid y_0)) \\ &\le \sqrt{2} B(\theta, y_0) \sqrt{\mathrm{KL}(q(z) \parallel p_\theta(z \mid y_0))},\end{aligned}$$

where TV denotes the total variation distance of probability measures: $\mathrm{TV}(p, q) = \frac{1}{2}\int |q(x) - p(x)| dx$, and the last inequality is due to $\mathrm{TV}(q, p) \le \sqrt{\frac{1}{2}\mathrm{KL}(q \parallel p)}$. □

In the context of the diffusion model, the latent variable $z = y_{1:T}$ is the diffusion trajectory, $q(y_{1:T})$ is the distribution over the diffusion trajectory $y_{1:T}$, and $p_\theta(y_{1:T}|y_0)$ is the corresponding diffusion reverse process. Thus, whenever the variational approximation is good (small KL), the ELBO gradient is provably close to the true score.

Empirical evidence for ELBO gradient approximation

Assume our model is $\theta = (A, B, c)$, and the noise is predicted by $\epsilon_\theta(y_t, t, x) = A_t y_t + B_t x + c_t$. True data $y \sim \mathcal{N}(Wx, I)$. In this way, there is a closed-form solution for true $\nabla_\theta \log p(y_0|x)$ since y_0 is a Gaussian and $y_t = \sqrt{\bar{\alpha}_t} y_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$.

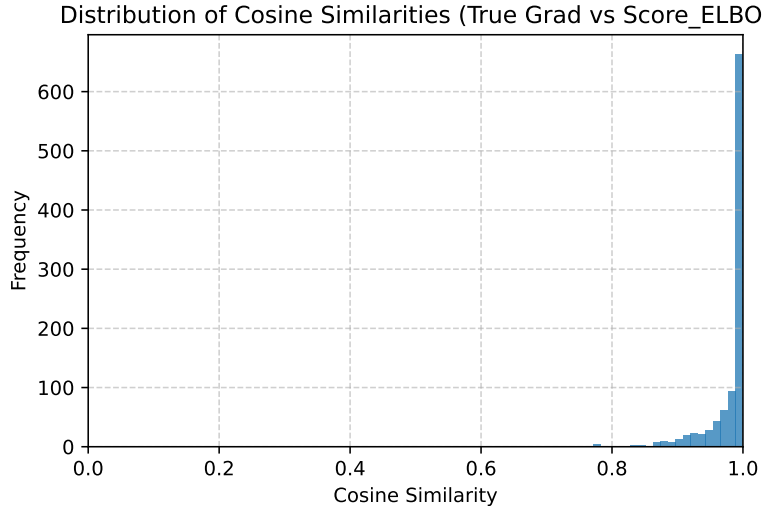


Figure 5.6: Cosine similarity between the true gradient and the estimated gradient using a linear model.

Deterministic Optimization and Gaussian Model in Stochastic Optimization

Deterministic Optimization Since deterministic can also be viewed as a reparameterization trick without any randomness ϵ, we can reuse our derivation in Section 5.4 and compute the gradient $\nabla_x F(x)$ by Eq. 5.5.

Gaussian Model in Stochastic Optimization Since there are many recent papers that use the Gaussian model as a predictor for stochastic DFL, we also claim that it has the reparameterization and score function form.

1. Reparameterization with Gaussian Model. Also using the reparameterization trick:

$$\nabla_\theta \mathbb{E}_{y \sim P_\theta(\cdot|x)}[f(y,z)] = \mathbb{E}_{\epsilon \sim P(\epsilon)}[\nabla_\theta y^\top \nabla_y f(y,z)].$$

   But with the predictor instantiated as a Gaussian model, i.e., y is computed by $y = R(\epsilon, \theta|x))^\top$ and R is a Gaussian model.

2. Score Function with Gaussian Model. Recall we need to approximate the term $\log P_\theta(y|x)$ for the diffusion model. However, this term has a closed-form

for a Gaussian model. Assume our Gaussian model is $\mathcal{N}(\mu_\theta, \Sigma_\theta)$, then the negative log-likelihood is

$$\log P_\theta(y \mid x) = -\frac{1}{2}[(y-\mu_\theta)^\top \Sigma_\theta^{-1}(y-\mu_\theta) + \log\det\Sigma_\theta + d\log(2\pi)].$$

Then, the gradient for the score function can be calculated using $\nabla_\theta \log P_\theta(y \mid x)$. Gaussian models are powerful tools for many DFL tasks. However, we want to claim that the diffusion model is more general and requires less model tuning and model assumptions.

Connections with Offline Contextual Bandit Methods

DFL and offline contextual bandits both involve learning from contextual features to make decisions, but they differ in key assumptions and objectives. Specifically, in DFL, the decision z is a solution of solving a known optimization problem, with explicit constraints and objective (e.g., quadratic objective with convex constraints). In other words, DFL explicitly incorporates the known structure of the decision task into the learning problem. Predict-then-optimize methods specifically aim to leverage this structure of the optimization problem to learn better predictions under fewer samples. This incorporation of task structure is the key motivation for predict-then-optimize methods.

In an offline bandit view, each decision (action) z is obtained by a policy that is learned directly without using the fact that z comes from solving a particular optimization problem. In theory, if the model is expressive enough, an offline bandit algorithm can learn the decision with enough samples, but it may ignore the underlying optimization structure and thus require more samples.

Here, we implement an offline full-information (the true cost function is revealed) contextual bandit method using policy-based learning methods. Specifically, we first sample a batch B from the offline dataset, and then compute the decisions (actions) by a policy network ϕ. Our goal is to train the policy network ϕ to minimize the following loss function:

$$L(\phi) = \frac{1}{|B|}\sum_{(x,y)\in B} f(\phi(x), y).$$

After training T epochs, we evaluate ϕ in a test set and report the results.

Training Details and Hyperparameters

We summarize our model settings for the deterministic DFL, Gaussian DFL, and diffusion DFL in Table 5.2.

Table 5.2: Architectural and training differences among deterministic, Gaussian, and diffusion-based DFL methods.

Parameter	**Deterministic MLP**	**Gaussian MLP**	**Diffusion Model**
Model Architecture			
Trunk (layers × width)	2×1024	2×1024	2×1024
Activation	ReLU / SiLU	ReLU / SiLU	SiLU
Inputs	$x \in \mathbb{R}^{d_x}$	$x \in \mathbb{R}^{d_x}$	$[y_t,\ t,\ x]$
Time embedding	—	—	Sinusoidal t (16-d) $\rightarrow$ 2 FC + SiLU
Output head	$\hat{y} \in \mathbb{R}^{d_y}$	$\mu(x),\ \log \sigma^2(x)$	$\hat{\epsilon}_\theta(y_t, t, x)$
Uncertainty	None (point)	Gaussian $\mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$	Diffusion process $P_\theta(y \mid x)$
Training			
Loss	MSE(y)	Gaussian NLL	Weighted denoising MSE
DFL gradient	Implicit diff. via KKT	(1) Reparam (Gaussian) (2) Gaussian score	(1) Reparam (Diffusion) (2) Weighted-ELBO score
Sample size M (synthetic / power / portfolio)	—	10/25/50	10/25/50
Learning rate	1×10^{-4} (typical)	Reparam: 1×10^{-5} Score-fn: 8×10^{-6}	Reparam: 1×10^{-5} Score-fn: 8×10^{-6}
Inference			
Procedure	Use $\hat{y}$ directly	Sample M outputs $y^{(m)}$	Reverse diffusion to get M samples $y^{(m)}$

For all experiments, we perform 10 random seeds to evaluate variability.

Implementation Details Stochastic optimization introduces two main computational bottlenecks: (i) sampling multiple Monte Carlo instances from the diffusion model, and (ii) repeatedly solving the downstream optimization problem. We address both in our implementation:

- **Parallel diffusion sampling.** We sample in parallel across many noise realizations, which makes efficient use of hardware and reduces the per-sample overhead. In our supplementary material, for each task (`power_sched`, `stock_portfolio`, `synthetic_example`), the file `{task}/diffusion_opt.py` implements the function `sample_elbo`, which accepts batched inputs and runs the diffusion sampler for the entire batch in parallel. This allows us to draw many Monte Carlo samples in *one single forward pass*.
- **Parallel optimization solving.** The downstream optimization solver is also executed in parallel across different samples. In `{task}/cvxpy_{task}.py`, the function `cvxpy_{task}_parallel_kkt` constructs a batched KKT system for the Monte Carlo samples and *solves these batched KKT systems in parallel* (across CVXPY worker jobs).

Details of synthetic task

In this example, we consider a factory that decides how much to manufacture for each of $d \in \mathbb{N}$ products. The parameter $Y \in \mathbb{R}^d$ represents the *profit margin* for each product, i.e., Y_i is the profit per unit of product i; due to uncertainty in market conditions, Y is uncertain. The factory's decision $z \in [0, C]^d$ represents how much of each product to manufacture, where C is the maximum capacity for each product. For simplicity, we do not consider any contextual features x in this example. That means DFL learns a distribution that generates y that can minimize the decision objective.

Suppose that the factory has a risk-averse cost function $f(y, z) = \exp(-y^\top z)$, which indicates that the factory wants to put a larger weight on the product with higher profit Y_i. Intuitively, if the factory knew Y exactly, then the optimal strategy would be all-or-nothing: set $z_i = C$ if $Y_i > 0$, or $z_i = 0$ if $Y_i < 0$. Likewise, with respect to a point prediction of Y, the optimal deterministic decision $z^*_{\text{det}} \in \{0, C\}^d$ is attained on the boundary of the feasible set.

Under uncertainty, the decision-maker seeks to minimize the **expected cost** by solving a stochastic optimization problem:

$$z^*_{\text{sto}} \in \arg\min_{z\in[0,C]^d} \mathbb{E}_{y\sim P_\theta(\cdot|x)}[\exp(-y^\top z)].$$

In this stochastic case, the optimal investment z^*_{sto} typically lies in the interior of the feasible region, which balances the potential high reward of investing against the risk of losses.

Then, we compute the necessities for diffusion DFL:

$$\begin{aligned} H &= \mathbb{E}_{y\sim P_\theta(\cdot|x)}[\nabla^2_{zz} g(z^*, y)] + (\lambda^*)^\top \nabla^2_{zz} h(x, z^*) = \exp(-y^\top z) yy^\top \\ G &= \nabla_z h(x, z^*) = -\exp(-y^\top z) y. \end{aligned}$$

For reparameterization, we have

$$\begin{aligned} (\frac{dloss}{d\theta})^\top &= \begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \end{bmatrix}^\top \begin{bmatrix} \frac{\partial z^*}{\partial \theta} \\ \frac{\partial \lambda^*}{\partial \theta} \end{bmatrix} \\ &= -\begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top \\ D(\lambda^*)G & D(h(x,z^*)) \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{M}\sum_{i=1}^M (\nabla_\theta y_i)^\top \nabla^2_{zy} g(z^*, y_i) \\ 0 \end{bmatrix} \end{aligned}$$

where $\nabla^2_{zy} g(z^*, y) = \exp(-y^\top z)(yz^\top - I_d)$ in this case.

For the score function, we have

$$\begin{aligned} (\frac{dloss}{d\theta})^\top &= -\begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top \\ D(\lambda^*)G & D(h(x,z^*)) \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_y[\nabla_z g(z^*, y)(\frac{dELBO}{d\theta})^\top] \\ 0 \end{bmatrix} \\ &\approx -\begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top \\ D(\lambda^*)G & D(h(x,z^*)) \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{M}\sum_{i=1}^M \nabla_z g(z^*, y_i)(\frac{dELBO_i}{d\theta})^\top \\ 0 \end{bmatrix}. \end{aligned}$$

Details on power schedule task

This task involves a 24-hour electricity generation scheduling problem with uncertain demand. The decision $z \in \mathbb{R}^{24}$ represents the electricity output to schedule for each hour of the next day. The uncertainty $y \in \mathbb{R}^{24}$ represents the actual power demand for each of the 24 hours. The goal is to meet demand as closely as possible at minimum cost. We also consider a decision cost function that penalizes storage, excess generation, and ramping following [73]:

1. Let γ_s and γ_a be the per-unit costs of shortage (not meeting demand) and excess (over-generation), respectively. We use $\gamma_s = 50$ and $\gamma_e = 0.5$ in our experiment.

2. Let c_r be a penalty on hour-to-hour changes in generation. We use $c_r = 0.4$ in appropriate units.

Formally, if $z = (z_1, \ldots, z_{24})$ and $y = (y_1, \ldots, y_{24})$, the loss for a single day is

$$\min_z \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)] = \sum_{i=1}^{24} \mathbb{E}_{y\sim P_\theta(\cdot|x)}[\gamma_s[y_i - z_i]_+ + \gamma_e[z_i - y_i]_+ + \frac{1}{2}(z_i - y_i)^2]$$

$$\text{s.t. } |z_i - z_{i-1}| \le c_r \text{ for all } i \in \{1, 2, \ldots, 24\}.$$

Then, we compute the necessities for diffusion DFL:

$$H = \mathbb{E}_{y\sim P_\theta(\cdot|x)}[\nabla^2_{zz} g(z^*, y)] + (\lambda^*)^\top \nabla^2_{zz} h(x, z^*) = I_n$$

$$G = \nabla_z h(x, z^*) = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}.$$

For reparameterization, we have

$$\begin{aligned}(\frac{dloss}{d\theta})^\top &= \begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \end{bmatrix}^\top \begin{bmatrix} \frac{\partial z^*}{\partial \theta} \\ \frac{\partial \lambda^*}{\partial \theta} \end{bmatrix} \\ &= -\begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top \\ D(\lambda^*)G & D(h(x,z^*)) \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{M}\sum_{i=1}^M (\nabla_\theta y_i)^\top \nabla^2_{zy} g(z^*, y_i) \\ 0 \end{bmatrix}\end{aligned}$$

where $\nabla^2_{zy} g(z^*, y) = -I$ in this case.

For the score function, we have

$$\begin{aligned}(\frac{dloss}{d\theta})^\top &= -\begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top \\ D(\lambda^*)G & D(h(x,z^*)) \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_y[\nabla_z g(z^*, y)(\frac{dELBO}{d\theta})^\top] \\ 0 \end{bmatrix} \\ &\approx -\begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top \\ D(\lambda^*)G & D(h(x,z^*)) \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{M}\sum_{i=1}^M \nabla_z g(z^*, y_i)(\frac{dELBO_i}{d\theta})^\top \\ 0 \end{bmatrix}.\end{aligned}$$

Dataset. We use real, public historical electricity load data from the PJM regional grid [204]. The features x for each day include: the previous day's 24-hour load profile, the previous day's temperature profile, calendar features, and seasonal sinusoidal features. In total, $d_x = 150$ features for each day were constructed. We normalized all input features for training. The target label y is the next day's 24-hour load vector.

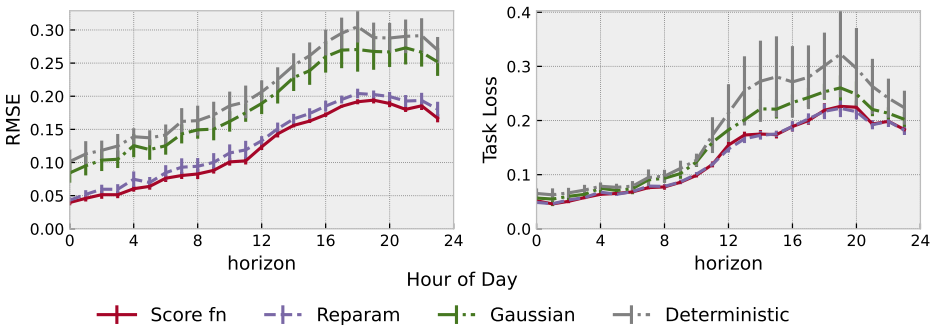


Figure 5.7: Results on the 24-hour power grid scheduling task.

For completeness, we include an extended comparison of different sample sizes in Figure 5.9, which further highlights that additional samples yield diminishing returns in accuracy while linearly increasing compute cost. We also find that adding a small regularizer during DFL training can help the model learning the data distribution and avoid some bad local minima, leading to a stable training process.

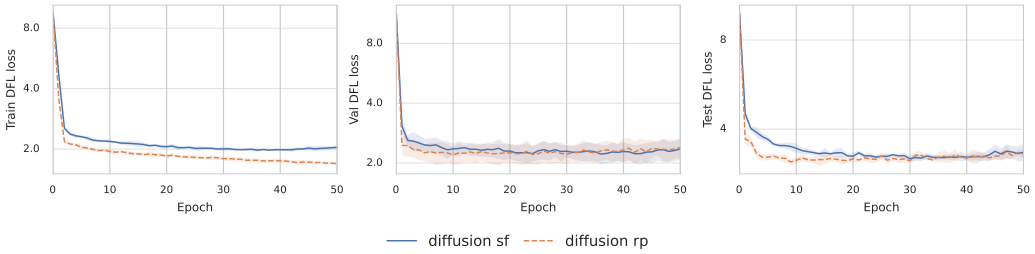


Figure 5.8: Train, validation, and test DFL loss for diffusion DFL using score function vs. reparameterization. Solid lines show the mean loss over multiple runs with different random seeds; shaded regions indicate standard deviation across runs.

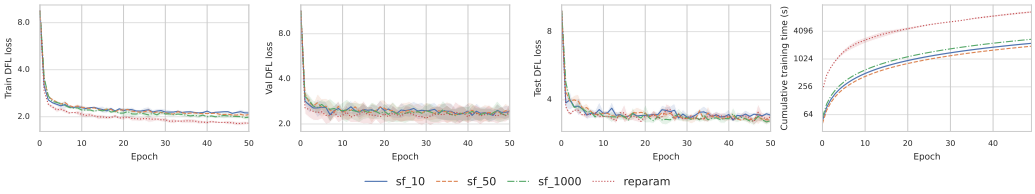


Figure 5.9: Comparison between different sample sizes for score function and reparameterization.

Details on stock portfolio task

We consider a mean-variance portfolio optimization problem with uncertain returns. Here $y \in \mathbb{R}^n$ represents the random next-day returns of n assets (stocks), and $z \in \mathbb{R}^n$ are the portfolio weights we assign to each asset (the fraction of our capital invested in each stock). Our goal is to maximize expected return while keeping the risk

(variance) low. This can be written as minimizing a loss that is a negative expected return plus a quadratic penalty on variance:

$$\min_z \ \mathbb{E}_{y\sim P_\theta(\cdot|x)}[f(y,z)] = \mathbb{E}_{y\sim P_\theta(\cdot|x)}\left[\frac{\alpha}{2} z^\top y y^\top z - y^\top z\right]$$
$$\text{s.t. } z^\top \mathbf{1} = 1,\ 0 \le z_i \le 1.$$

Then, we compute the necessities for diffusion DFL:

$$H = \mathbb{E}_{y\sim P_\theta(\cdot|x)}[\nabla^2_{zz} f(z^*, y)] + (\lambda^*)^\top \nabla^2_{zz} h(x, z^*) + \nabla^2_{zz}(\nu^{*\top} g(x, z^*)) = \alpha\, \mathbb{E}_{y\sim P_\theta(y|x)}[yy^\top],$$
$$G = \nabla_z h(x, z^*) = \begin{bmatrix} I_n, \\ -I_n \end{bmatrix}$$
$$Q = \nabla_z g(x, z^*) = \mathbf{1}^\top.$$

For reparameterization, we have

$$\begin{aligned}(\frac{dloss}{d\theta})^\top &= \begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} \frac{\partial z^*}{\partial \theta} \\ \frac{\partial \lambda^*}{\partial \theta} \\ \frac{\partial \nu^*}{\partial \theta} \end{bmatrix} \\ &\approx -\begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x,z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{M}\sum_{i=1}^M (\nabla_\theta y_i)^\top \nabla^2_{zy} g(z^*, y_i) \\ 0 \\ 0 \end{bmatrix}\end{aligned}$$

where $\nabla^2_{zy} f(z^*, y) = \mathbb{E}_y[2\alpha y^\top z - 1]$ in this case.

For the score function, we have

$$\begin{aligned}(\frac{dloss}{d\theta})^\top &= \begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x,z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{E}_y[\nabla_z g(z^*, y)(\frac{dELBO}{d\theta})^\top] \\ 0 \\ 0 \end{bmatrix} \\ &\approx -\begin{bmatrix} \frac{dloss}{dz^*} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} H & G^\top & Q^\top \\ D(\lambda^*)G & D(h(x,z^*)) & 0 \\ Q & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{M}\sum_{i=1}^M \nabla_z g(z^*, y_i)(\frac{dELBO_i}{d\theta})^\top \\ 0 \\ 0 \end{bmatrix},\end{aligned}$$

where $\nabla_z g(z^*, y_i) = \alpha y_i y_i^\top z^* - y_i$.

Dataset. We use daily stock prices from 2004–2017 for constituents of the S&P 500 index [212]. We obtained this data via Quandl's API (specifically WIKI pricing data; the user will need a Quandl API key to replicate. We compute daily returns for each stock (percentage change). To construct features x, we use a rolling window of

recent history for each asset. Specifically, for each day, a data point is for predicting next day's returns: we include the past 5 days of returns for each of the n assets, past 5 days of trading volume for each asset, plus some aggregate features. To avoid an explosion of dimension with large n, we also include PCA-compressed features: we take the top principal components of the last 5-day return matrix to summarize cross-asset trends. In the end, for $n = 50$ assets, we ended up with $d_x = 28$ features. All features are normalized and we use a time-series split: first 70% of days for training (2004–2013), next 10% for validation (2014), last 20% for test (2015–2017). We evaluate performance on the test set by simulating the portfolio selection every day and computing the average return achieved.

(Additional task) Details on inventory stock problem

We also validate our approaches on a toy inventory control problem. In this task, the uncertain demand y is drawn from a multi-modal distribution (a mixture of Gaussians), where we vary the number of mixture components K to control the distribution complexity:

$$p(x) = \sum_{j=1}^{K} \pi_j \phi(x; \mu_j, \Sigma_j),$$

where π_j is the probability of choosing component j and $\phi(x; \mu_j, \Sigma_j)$ is a multivariate Gaussian density with parameter (μ_j, Σ_j). The cost function follows the standard newsvendor formulation with piecewise penalties for under-stock and over-stock:

$$f(y, z) = c_0 z + \frac{1}{2} q_0 z^2 + c_b [y - z]_+ + \frac{1}{3} r_b ([y - z]_+^3) + c_h [z - y]_+ + \frac{1}{3} r_h ([z - y]_+^3).$$

Our learning objective is to minimize the expected cost over this stochastic demand, i.e., a stochastic optimization problem:

$$\min_z L(\theta) = \mathbb{E}_{y \sim P(\cdot|x)}[f(y, z)] \quad \text{s.t. } 0 \le z \le z_{max}.$$

We compare a deterministic DFL model against our diffusion DFL model on this toy task. Figure 5.10 summarizes the results, where the diffusion model (stochastic DFL) achieves substantially lower regret on average than the deterministic model. Besides, we observe that the decision z obtained by our diffusion method closely tracks the true optimal decisions z^* by capturing the multi-modal demand uncertainty, whereas the deterministic predictor's decisions deviate more. In Figure 5.10 (c), we directly

Figure 5.10: Toy decision task comparing deterministic and diffusion DFL. *Left*: distribution of per-instance regret (lower is better). *Middle*: distribution of chosen decision z in the lower-level; the stochastic method tracks the true distribution z^* more closely. *Right*: pairwise win-rate on test set; a large fraction of costs from the stochastic method are lower than the deterministic one, indicating that modeling uncertainty yields better decisions.

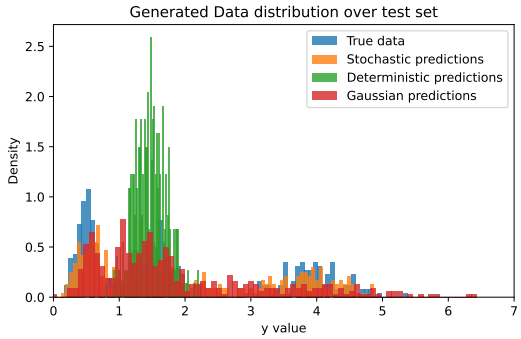


Figure 5.11: Prediction distribution for inventory stock problem.

compare the decision outcomes via a win-rate: the fraction of test instances where one method achieves a lower cost than the other. The diffusion DFL method attains a win-rate of about 75% against the deterministic baseline, which confirms that modeling uncertainty leads to better downstream decisions

Additional Related Works and Discussions

Stochastic optimization Making decisions under uncertainty is a classic topic in operations research and machine learning [235]. Stochastic optimization formulations explicitly consider uncertainty by optimizing the expected objective over a distribution of unknown parameters. A common approach is the Sample Average Approximation (SAA) [136, 14, 287], which draws many samples from the estimated distribution and solves an approximated deterministic problem minimizing the average cost. While SAA can handle arbitrary uncertainty distributions in theory, it becomes very computationally expensive and still does not consider the distribution during optimization [130]. It will lead to optimizing the *sample mean*, which may yield a

decision that performs poorly if reality often falls into one of several distinct models far from the mean [130, 78].

When is Diffusion-DFL useful? Our experiments suggest that Diffusion DFL will be especially useful in decision-making settings where the uncertain parameter in the objective is high-dimensional, has a multimodal distribution, and limited training data is available. We believe that promising directions of future work include accommodating uncertainty in constraints in addition to the objective function, investigating the trade-off between model calibration and decision-making quality, and adapting Diffusion DFL to online decision-making settings.

Offline Contextual Bandits (OCB) Offline contextual bandit approaches provide an alternative way to learn decisions from contextual features by directly optimizing a policy that maps each context to an action [2, 187, 41, 89, 227]. A key difficulty is that feedback is typically *partial*: the log reveals rewards only for actions taken by a behavior policy, so learning and evaluation rely on off-policy estimators and assumptions such as overlap between the learned and behavior policies. Recent OCB methods address this challenge via optimistic or pessimistic objectives [187, 283], variance control [155, 154, 286], conservative regularization [261], and robust optimization [305] to avoid out-of-distribution actions. In all these methods, the policy is learned purely from logged data, without using the fact that the decision comes from solving a particular optimization problem. This is fundamentally different from DFL in terms of the prior knowledge and optimization structure. In DFL, the decision z is a solution of solving a known optimization problem, with explicit constraints and objectives. In this chapter, we empirically show that Diffusion-DFL achieves better performance than the policy-based OCB method.

Part II

Online and Reinforcement Learning

Chapter 6

ONLINE LEARNING FOR ROBUST VOLTAGE CONTROL UNDER UNCERTAIN GRID TOPOLOGY

In this chapter, we investigate the problem of online learning for robust voltage control in power distribution systems under uncertainty in grid topology. The goal is to design an online learning algorithm that can adaptively adjust voltage control actions while accounting for uncertainties in the grid topology arising from line outages or changes in automatic switches. Unlike previous chapters, this chapter does not use end-to-end learning approaches, or even gradient-based optimization methods. Instead, this chapter builds upon the framework of online convex optimization (OCO) and shows how OCO can be used to learn uncertainty sets for a robust controller in the context of the voltage control task.

This chapter is primarily based on the following two papers, both of which are licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0)[1]:

[307] Christopher Yeh, Jing Yu, Yuanyuan Shi, and Adam Wierman. 2024. Online learning for robust voltage control under uncertain grid topology. *IEEE Transactions on Smart Grid*, 15, 5, (Sept. 2024), 4754–4764. doi:10.1109/TSG.2024.3383804.

[308] Christopher Yeh, Jing Yu, Yuanyuan Shi, and Adam Wierman. 2022. Robust online voltage control with an unknown grid topology. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*. Association for Computing Machinery, (June 2022), 240–250. ISBN: 9781450393973. doi:10.1145/3538637.3538853.

6.1 Introduction

Operators of electricity distribution grids must maintain voltages at each bus within certain operating limits, as deviations from such limits may damage electrical equipment and cause power outages [95, 103]. This "voltage control" or "voltage regulation" problem has been well-studied, *e.g.*, [219, 328, 181] and the references therein. Voltage control devices and algorithms aim to guarantee grid stability and minimize the costs associated with control inputs. While classic voltage regulation devices such as tap-changing transformers are effective in dealing with *slow* voltage variations [232, 91], increasing penetration of renewables leads to faster variations, and a growing body of literature has focused on inverter-based controllers that

[1] https://creativecommons.org/licenses/by/4.0/

can respond quickly by adjusting their active and reactive power set-points. Most of these works cast voltage control as an optimization problem and then propose different centralized or decentralized algorithms depending on the communication infrastructure.

Typically, voltage control algorithms assume *exact knowledge* of the underlying grid topology. This includes centralized controllers such as algorithms based on model predictive control (MPC) which optimize control decisions for a short-term horizon. [101] uses MPC to manage distributed generation and energy storage systems, whereas [173] proposes a robust MPC controller that is robust to uncertainty in the forecasts of future loads and solar generation.

However, the exact grid topology and line parameters are often not known, and using existing voltage control algorithms with incorrect grid information may lead to problems with grid stability [200, 153]. For example, parts of the grid may undergo reconfiguration due to load balancing or unplanned maintenance, as frequently as every hour of the day [63, 159, 65, 80]. This problem is exacerbated by the increasing integration of distributed energy resources (DERs), such as photovoltaic (PV) and storage devices. Especially in distribution grids, where DERs are not owned or operated by the electricity utility, the grid operator may lack up-to-date information about the grid topology [162]. While a grid operator can install sensors to help identify the current network topology, unless such sensors are densely deployed (at great cost), uncertainty about the topology remains. Thus, distribution grid operators cannot expect to operate with perfect topology information and the design of voltage control algorithms robust to unknown grid topology is crucial.

There are several families of existing algorithms that do not require knowing the network topology: decentralized controllers, model-free controllers, and controllers that first try to infer the network topology. While decentralized voltage control algorithms are generally efficient to implement, such controllers lack voltage stability guarantees when the load is time-varying [39, 156, 328, 264, 210]. Likewise, model-free controllers based on deep reinforcement learning do not require knowing the network topology, but they generally have no performance or voltage stability guarantees and are therefore not suitable for safety-critical infrastructure [74, 285, 297, 94, 257]. Some recent works [243, 85, 62] have proposed methods for introducing stability guarantees for model-free deep reinforcement learning approaches. Their main tool is Lyapunov stability theory, from which a structural constraint for stable controllers is derived, and policy optimization with the constraint is performed.

However, their stability guarantees are only valid over an infinite time horizon, and achieving good performance with deep reinforcement learning generally requires large amounts of historical training data. In contrast, our proposed framework jointly learns the system model (consistent with data) and stable controller in an online fashion, achieving a finite-mistake guarantee and good performance without relying on historical data.

Another standard approach for handling uncertainty about network topology is to first estimate the topology and line parameters using a form of system identification with data and then apply a standard voltage control algorithm using the identified network topology. There is a growing literature of such data-driven methods, *e.g.*, [129, 162, 200, 153, 81, 43, 51, 12, 296, 191, 66, 53]. A common approach is to leverage least squares for system model estimation. The estimation and therefore control guarantees depend on statistical modeling of measurement noise (*e.g.*, Gaussian). In contrast, we leverage online learning in order to be robust against any bounded disturbances, such as modeling errors and adversarial noise. While least squares-based algorithms focus on *asymptotic* estimation convergence, *e.g.* [44, 313], we present a *finite* mistake guarantee that is crucial for safe *transient* system behavior.

Another prominent approach is to use graphical models for topology reconstruction [64], via maximum likelihood methods while enforcing other structural restrictions like low-rank and sparsity. However, these methods that first perform some form of system identification have drawbacks. First, the estimated topology and/or system dynamics may be imperfect [238], and applying standard voltage control algorithms using these imperfect estimates may still lead to system instability. Second, these methods either assume access to historical data or require acquiring data online over hundreds of time steps, during which the stability of the system is ignored [162, 64]. In contrast, our proposed approach does not perform system identification separately from control; the joint operation of our robust controller with the system dynamics estimation gives rise to our stability guarantee.

Contributions

We propose a new approach for voltage control over an uncertain grid topology that does not perform system identification and voltage control separately. Instead, our approach robustly learns to stabilize voltage within the desired limits directly, *without prior knowledge of the topology and without needing to precisely learn the topology*.

Our approach takes ideas from online nested convex body chasing (NCBC) [13] and

robust predictive control and combines them using a new learning framework [111] to design a voltage control algorithm. Intuitively, we use a NCBC algorithm to track the set of topologies that are consistent with the observed voltage measurements—as more measurements are taken, the set of consistent topologies shrinks (and so the sets are nested). As these measurements are taken, a form of robust predictive control is used for voltage control, where the robustness guarantee is used to ensure that the uncertainty about the topology can be handled. Our main result (Theorem 7) provides a finite error stability bound for the overall controller, which is summarized in Algorithm 6. This represents the first voltage control algorithm that is provably robust to large uncertainty about network topology.

In addition to providing theoretical guarantees, we demonstrate the effectiveness of our proposed approach using a case study of a 56-bus distribution grid from the Southern California Edison (SCE) utility [84]. In this setting, we give the controller no prior information about the topology of the grid, yet the controller quickly narrows down the set of topologies and line parameters that are consistent with its observations and adjusts reactive power generation to keep voltages within desired safety limits when faced with disturbance. In fact, our controller's performance nearly matches that of controllers which assume perfect knowledge of the topology, even when given only partial observations of bus voltages.

6.2 Model

We study voltage control on an unknown grid topology. We consider a radial (tree-structured) power distribution network represented as a connected directed graph $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{0, 1, 2, \ldots, n\}$ is the set of buses (nodes) and $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ is the set of lines (directed edges). Let the network be rooted at bus 0 (the substation or slack bus), and let other buses be branch buses. Let $\mathcal{C} \subseteq \mathcal{N}$ denote the subset of buses with controllable reactive power injection. Because the network is radial and rooted at bus 0, there is a unique path $\mathcal{P}_i$ from bus 0 to any other bus i. For branch buses, let $v \in \mathbb{R}^n$ be their squared voltage magnitudes and $p + \mathbf{i}q$ be their complex power injection, where $p \in \mathbb{R}^n$ (units W) is the net active power injection, and $q \in \mathbb{R}^n$ (units var) is the net reactive power injection. The DistFlow branch

equations [17] for a distribution grid are as follows, for all $j \in \mathcal{N}$ and $(i,j) \in \mathcal{E}$:

$$-p_j = P_{ij} - r_{ij} l_{ij} - \sum_{k:(j,k)\in\mathcal{E}} P_{jk} \tag{6.1a}$$

$$-q_j = Q_{ij} - x_{ij} l_{ij} - \sum_{k:(j,k)\in\mathcal{E}} Q_{jk} \tag{6.1b}$$

$$v_j = v_i - 2(r_{ij}P_{ij} + x_{ij}Q_{ij}) + (r_{ij}^2 + x_{ij}^2) l_{ij} \tag{6.1c}$$

$$l_{ij} = \frac{P_{ij}^2 + Q_{ij}^2}{v_i} \tag{6.1d}$$

where P_{ij} and Q_{ij} represent the active power and reactive power flow on line (i,j), and $r_{ij}, x_{ij} > 0$ are the real-valued line resistance and reactance (units Ω). Equations (6.1a) and (6.1b) represent the real and reactive power conservation at bus j, and (6.1c) represents the voltage drop from bus i to bus j.

Assuming the branch power losses $(r_{ij}l_{ij}, x_{ij}l_{ij})$ are negligible yields the simplified DistFlow equations [18], which can be rearranged into

$$v = R^\star p + X^\star q + v^0 \mathbf{1}_n \tag{6.2}$$

where $v^0 \in \mathbb{R}^n$ is the known, constant squared voltage magnitude at the substation, and $R^\star, X^\star \in \mathbb{S}^n$ are computed from the network topology and line parameters

$$R^\star_{ij} := 2\sum_{(h,k)\in\mathcal{P}_i\cap\mathcal{P}_j} r_{hk}, \quad X^\star_{ij} := 2\sum_{(h,k)\in\mathcal{P}_i\cap\mathcal{P}_j} x_{hk}, \quad i,j \in [n] \tag{6.3}$$

with $[n] := \{1,\dots,n\}$ [156]. ($\mathbb{S}^n$ is the set of symmetric $n \times n$ matrices.) $R^\star, X^\star$ are positive definite with nonnegative entries [83], and the largest entry of each row of these matrices is along the diagonal, since

$$X^\star_{ij} = 2\sum_{(h,k)\in\mathcal{P}_i\cap\mathcal{P}_j} x_{hk} \le 2\sum_{(h,k)\in\mathcal{P}_i} x_{hk} = X^\star_{ii} \tag{6.4}$$

and likewise for $R^\star_{ij} \le R^\star_{ii}$.

We assume that the active power injection p is exogenous but that reactive power at each bus can be decomposed as $q = q^c + q^e$, where q^c is the "controllable" component and q^e is the "exogenous" (*i.e.*, uncontrollable) component. Following [156], we define $v^{\text{par}} = R^\star p + X^\star q^e + v^0 \mathbf{1}_n \in \mathbb{R}^n$ ("par" stands for "partial") representing the exogenous effects on voltage. Then, $v = X^\star q^c + v^{\text{par}}$, which can be modeled as a discrete-time linear system

$$v(t+1) = X^\star q^c(t) + v^{\text{par}}(t). \tag{6.5}$$

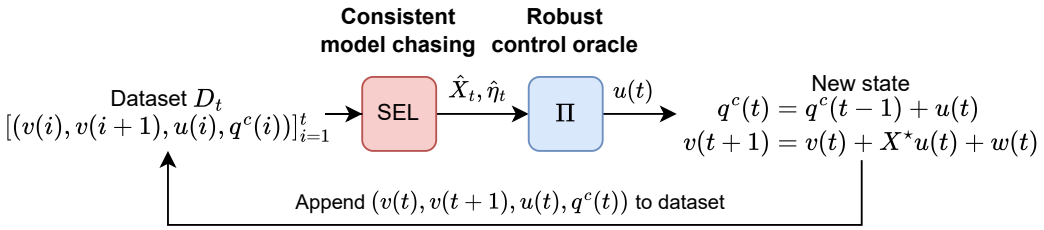


Figure 6.1: Online robust voltage control framework

Substituting $u(t) = q^c(t) - q^c(t-1)$ (change in controllable reactive power injection) and $w(t) = v^{\text{par}}(t) - v^{\text{par}}(t-1)$ (change in exogenous noise) yields the linear dynamical system

$$v(t+1) = v(t) + X^\star u(t) + w(t). \tag{6.6}$$

The voltage control problem [84] is to drive the squared voltage magnitudes of each bus from an initial state $v(1) \in \mathbb{R}^n$ into a given multi-dimensional interval $[\underline{v}, \overline{v}] \subset \mathbb{R}^n$; it is possible that $v(1)$ does not start within the interval due to some large initial disturbance. For all $t \geq 2$, the voltage control algorithm aims to maintain $v(t)$ within $[\underline{v}, \overline{v}]$, ideally as close as possible to a "nominal" value $v^{\text{nom}} \in [\underline{v}, \overline{v}]$, typically $v^{\text{nom}} = (\underline{v} + \overline{v})/2$. The cost for deviating from v^{nom} is measured by $\|v(t) - v^{\text{nom}}\|_{P_v}^2$ for some positive semidefinite matrix P_v, where $\|x\|_A^2 := x^\top A x$.

At each time step, buses may change their reactive power injection $q^c(t)$ in order to regulate the voltage close to v^{nom}. The reactive power injection (including $q^c(0)$) is limited within a given bound $[\underline{q}, \overline{q}] \subset \mathbb{R}^n$. Buses not in C do not have any ability to control the reactive power injection: $\forall i \notin C.\ \underline{q}_i = \overline{q}_i = 0$. We do not place any hard "ramp constraints" on $u(t)$. However, we impose a quadratic ramping cost $\|u(t)\|_{P_u}^2$ where P_u is a positive semidefinite matrix.

In summary, the voltage control problem is to determine an online sequence of reactive power injections $q^c(1), q^c(2), \ldots$ to drive voltages $v(t)$ to a desired interval $[\underline{v}, \overline{v}]$ while minimizing voltage violation and control costs $\|v(t) - v^{\text{nom}}\|_{P_v}^2 + \|u(t)\|_{P_u}^2$. In this work, we solve the voltage control problem in the setting where $X^\star$ is unknown.

6.3 Robust Online Voltage Control

In this section we introduce our robust online voltage control algorithm (Algorithm 6) and its performance bound (Theorem 7), which is the main result of this chapter.

Algorithm

As shown in Figure 6.1, the algorithm has two main components: a consistent model chasing algorithm SEL (Algorithm 6, step 2) and a robust control oracle Π (Algorithm 6, step 3). SEL and Π are combined by adapting ideas from [111].

The model chasing algorithm SEL selects a consistent model for the robust control oracle Π out of all plausible models that are consistent with the online observations and prior knowledge of the grid. The selection may use any competitive NCBC algorithm, which is the online problem of choosing a sequence of points within sequentially nested convex sets, with the aim of minimizing the sum of distances between the chosen points [13]. In our experiments, we use a simple projection-based NCBC algorithm, detailed in Section 6.5.

The robust control oracle Π is a novel robust predictive controller (Theorem 9). The robustness guarantee of Π is necessary for the analysis which integrates SEL with Π to provide the finite mistake guarantee of the overall algorithm. We remark that other choices for either component are possible, as long as they provide the guarantees needed in the analysis in Section 6.4.

Intuitively, SEL and Π are combined in a way such that SEL always reduces the uncertainty about the unknown model whenever Π outputs an action that causes a voltage limit violation. This means that Π cannot take too many "bad" actions before the system uncertainty is small.

Assumptions

Before presenting the main results, we introduce three assumptions that underlie our analysis and discuss why they are both needed and practical.

Assumption 9. *The change in noise is bounded as*

$$\forall t: \quad \|w(t)\|_\infty \leq \eta^\star,$$

where $w(t) = v^{\text{par}}(t) - v^{\text{par}}(t-1)$. $\eta^\star \in [0, \overline{\eta}]$ *is a constant (possibly unknown), while* $\overline{\eta}$ *is a known upper-bound.*

This first assumption is standard and bounds the noise in the dynamics. It represents realistic behavior in power systems where the active and exogenous reactive power injections do not vary dramatically between time steps, as can be seen by expanding

$w(t)$:

$$\begin{aligned} w(t) &= v^{\text{par}}(t) - v^{\text{par}}(t-1) \\ &= R^\star(p(t) - p(t-1)) + X^\star(q^e(t) - q^e(t-1)). \end{aligned}$$

For example, if the net active and exogenous reactive power injection is the same at time steps t and $t-1$, then $w(t) = 0$.

An unknown $\eta^\star$ indicates uncertainty in the maximum variability of the exogenous power injections. Unlike [308] which assumes a fixed η, our inclusion of both an unknown $\eta^\star$ and a known upper-bound $\overline{\eta}$ allows more flexibility in our algorithmic design and the incorporation of prior knowledge.

Assumption 10. *The true model $X^\star$ lies within a known compact, convex uncertainty set $\mathcal{X} \subset \mathbb{S}^n_+ \cap \mathbb{R}^{n\times n}_+$. ($\mathbb{S}^n_+$ is the set of $n \times n$ positive semidefinite matrices, and $\mathbb{R}^{n\times n}_+$ is the set of $n \times n$ matrices with nonnegative entries.)*

Our second assumption bounds the uncertainty about the network topology and line parameters. It ensures that the unknown true model parameters $X^\star$ belong to a compact, convex set $\mathcal{X}$, which is a minimal assumption necessary for proving an analytic guarantee. $P_1 = \mathcal{X} \times [0, \overline{\eta}]$ forms the initial "consistent set" (see Definition 6) for our consistent model chasing algorithm `SEL`.

This assumption is realistic, as a grid operator should have at least some prior knowledge about the distribution grid topology and the range of possible line parameters, even if they do not have the exact values. In cases where the grid has multiple possible topologies due to switches, $\mathcal{X}$ could be set to the convex hull of the corresponding X matrices.

Definition 5 ($\|\cdot\|_\triangle$ and $\|\cdot\|_{\triangle,\delta}$)**.** *For any matrix $X \in \mathbb{S}^n$ and scalars $\eta, \delta \geq 0$, define*

$$\|X\|_\triangle := \|\text{vech}(X)\|_2 = \sqrt{\sum_{i=1}^{n}\sum_{j=i}^{n} X_{ij}^2}$$

$$\|(X,\eta)\|_{\triangle,\delta} := \sqrt{\delta^2\eta^2 + \sum_{i=1}^{n}\sum_{j=i}^{n} X_{ij}^2} = \sqrt{\delta^2\eta^2 + \|X\|_\triangle^2},$$

where $\text{vech}(X)$ *is the half-vectorization of X defined as*

$$\text{vech}(X) = [X_{11}, X_{12}, \ldots, X_{1n}, X_{22}, X_{23}, \ldots, X_{2n}, \ldots, X_{nn}]^\top \in \mathbb{R}^{n(n+1)/2}.$$

For any sets $\mathcal{X} \subseteq \mathbb{S}^n$ and $A \subseteq \mathbb{R}$, we define diameters $\text{diam}(\mathcal{X})$ *and* $\text{diam}(\mathcal{X} \times A)$ *with respect to the norms $\|\cdot\|_\triangle$ and $\|\cdot\|_{\triangle,\delta}$, respectively.*

These norms isometrically map our parameter space to Euclidean space, enabling us to take advantage of known results on NCBC within Euclidean space. For the norm $\|\cdot\|_{\triangle,\delta}$, the hyperparameter δ trades off the weight between X and η in the norm. The choice of δ is discussed in Section 6.5.

In practice, we consider uncertainty sets of the form

$$\mathcal{X}_\alpha = \left\{ X \in \mathbb{S}^n_+ \cap \mathbb{R}^{n\times n}_+ \;\middle|\; \begin{array}{l} \|X - X^\star\|_\triangle \le \alpha \left\|X^\star\right\|_\triangle, \\ \forall i,j \in [n] : X_{ij} \le X_{ii} \end{array} \right\}$$

with $\operatorname{diam}(\mathcal{X}_\alpha) = 2\alpha \left\|X^\star\right\|_\triangle$. A larger α yields a larger uncertainty set. From Section 6.2 (*e.g.*, (6.4)), we know that $X^\star \in \mathcal{X}_\alpha$.

Furthermore, we can incorporate partial knowledge we may have of the network topology and/or line parameters by adding constraints to the description of $\mathcal{X}$. For example, if we know that the lowest common ancestor between buses i, j in the network is bus k, then we can add the following constraint on X, which is a consequence of (6.3):

$$X_{ij} = \begin{cases} 0, & k = 0 \\ X_{kk}, & \text{otherwise.} \end{cases} \tag{6.7}$$

If we additionally know the values for some line parameters x_{ij}, we may be able to further constrain some entries of X, again by applying (6.3).

Assumption 11. *There exists a compact, convex set $\mathcal{V}^{\text{par}} \subset \mathbb{R}^n$ such that $\forall t \ge 0$: $v^{\text{par}}(t) \in \mathcal{V}^{\text{par}}$. Furthermore, for some known $\epsilon > 0$,*

$$\begin{aligned} &\forall v^{\text{par}} \in \mathcal{V}^{\text{par}},\ X \in \mathcal{X}. \\ &\exists q^c \in [\underline{q}, \overline{q}] \text{ s.t. } Xq^c + v^{\text{par}} \in [\underline{v} + (\overline{\eta} + \epsilon)\mathbf{1},\ \overline{v} - (\overline{\eta} + \epsilon)\mathbf{1}]. \end{aligned}$$

Our final assumption is about the existence of feasible control actions for the robust control oracle. This assumption can be interpreted as either a bound on the noise, or a requirement that the controllable reactive power injection be flexible enough to satisfy the demand of any admissible noise. It represents the reasonable assumption that a grid operator should have installed enough controllable reactive power injection capability to perform voltage control. Intuitively, the $\overline{\eta}$ padding is required for robustness to the noise $w(t)$, while the ϵ padding is required for robustness to model uncertainty (*i.e.*, uncertainty about $X^\star$).

Algorithm 6 Online Robust Voltage Controller

Inputs

- desired nominal squared voltage magnitude: $v^{\text{nom}} \in \mathbb{R}^n$
- limits on the squared voltage magnitude: $[\underline{v}, \overline{v}] \subset \mathbb{R}^n$
- limits on the reactive power injection: $[\underline{q}, \overline{q}] \subset \mathbb{R}^n$
- initial state: $v(1), q^c(0) \in \mathbb{R}^n$
- state and action cost matrices: $P_v, P_u \in \mathbb{S}^n_+$
- compact convex uncertainty set for the model parameter: $\mathcal{X} \subset \mathbb{S}^n_+ \cap \mathbb{R}^{n\times n}_+$
- compact convex uncertainty set for exogenous voltage quantities: $\mathcal{V}^{\text{par}} \subset \mathbb{R}^n$
- upper bound for noise: $\overline{\eta} > 0$
- robustness padding: $\epsilon > 0$
- weight for slack variable: $\beta > 0$
- weight for noise accuracy: $\delta > 0$

Procedure

1. Initialize an empty trajectory $D_0 = [\,]$. Set $t = 1$.
2. Query the model chasing algorithm for a new consistent parameter estimate: $(\hat{X}_t, \hat{\eta}_t) \leftarrow \texttt{SEL}[D_t]$.

$$\texttt{SEL}[D_t] := \text{NCBC}(P_t, \hat{X}_{t-1}, \hat{\eta}_{t-1}) \tag{6.8a}$$

$$P_t := \left\{ (\hat{X}, \hat{\eta}) \;\middle|\; \begin{array}{l} \hat{X} \in \mathcal{X},\ \hat{\eta} \in [0, \overline{\eta}] \\ \forall (v_i, v_{i+1}, u_i, q_i^c) \in D_t : \\ \quad \|v_{i+1} - v_i - \hat{X} u_i\|_\infty \le \hat{\eta} \\ \quad v_{i+1} - \hat{X} q_i^c \in \mathcal{V}^{\text{par}} \end{array} \right\} \tag{6.8b}$$

3. Query the robust control oracle for the next control action: $u(t) \leftarrow \Pi_{\hat{X}_t, \hat{\eta}_t}(v(t))$.

$$\Pi_{\hat{X}_t, \hat{\eta}_t}(v(t)) := \operatorname*{arg\,min}_{u \in \mathbb{R}^n} \min_{\xi \in \mathbb{R}_+} \ \|\hat{v}' - v^{\text{nom}}\|^2_{P_v} + \|u\|^2_{P_u} + \beta \xi^2 \tag{6.9a}$$

$$\text{s.t. } \underline{q} \preceq q^c(t-1) + u \preceq \overline{q} \tag{6.9b}$$

$$\hat{v}' = v(t) + \hat{X}_t u \tag{6.9c}$$

$$k = \hat{\eta}_t + \rho \left(\frac{1}{\delta} + \|u\|_2 \right) \tag{6.9d}$$

$$\underline{v} + (k - \xi)\mathbf{1} \preceq \hat{v}' \preceq \overline{v} - (k - \xi)\mathbf{1} \tag{6.9e}$$

where $\rho = \delta\epsilon / (1 + \delta \|\overline{q} - \underline{q}\|_2)$.

4. Apply the control action $u(t)$. Observe the system transition to $v(t+1) = v(t) + X^\star u(t) + w(t)$ and $q^c(t) = q^c(t-1) + u(t)$.
5. Append $(v(t), v(t+1), u(t), q^c(t))$ to the trajectory:

$$D_t = [(v(i), v(i+1), u(i), q^c(i))]_{i=1}^t .$$

6. Increment $t \leftarrow t + 1$. Repeat from Step (2).

Main result

We now state our main result, which is a finite-error bound for Algorithm 6.

Theorem 7 (Main Result)**.** *Under Assumptions 9 to 11, Algorithm 6 ensures that the voltage limits will be violated at most* $\frac{2\gamma(m)}{\rho}\operatorname{diam}(\mathcal{X}\times[0,\overline{\eta}])+1$ *times, where* $\rho=\frac{\delta\epsilon}{1+\delta\|\overline{q}-\underline{q}\|_2}$ *and* $\gamma(m)$ *is the competitive ratio of the NCBC algorithm in* m*-dimensional Euclidean space, where* $m=1+\frac{n(n+1)}{2}$.

If $\eta^\star$ *is known, then the voltage limits will be violated at most* $\frac{2\gamma(m)}{\rho}\operatorname{diam}(\mathcal{X})+1$ *times, where* $\rho=\frac{\epsilon}{\|\overline{q}-\underline{q}\|_2}$ *and* $m=\frac{n(n+1)}{2}$.

To the best of our knowledge, this result is the first provable stability bound for voltage control in a setting where the network topology is unknown. It highlights that Algorithm 6 can ensure stability even after *unknown* changes to the network topology, *e.g.*, due to maintenance, failures, etc., without the need to perform system identification while remaining robust to any bounded and potentially adversarial perturbations satisfying Assumptions 9 and 11.

Intuitively, this result guarantees that the model chasing algorithm SEL will learn a "good enough" model for control quickly. When the robust controller Π makes a mistake, the model chasing algorithm will learn from that mistake and significantly reduce the set of consistent models. Because the initial set of consistent models is bounded, and this set shrinks a significant amount after each mistake, the total number of mistakes is bounded. Note that this finite mistake bound implies finite-time convergence to safe voltage limits without an explicit finite-time bound.

To interpret the error bounds in Theorem 7, we notice that they are proportional to the diameter of the parameter space and the competitive ratio $\gamma(m)$ of the NCBC algorithm, and inversely proportional to the oracle robustness margin ρ. Because of computational tractability concerns, our experiments implement SEL with a greedy projection-based NCBC algorithm with $\gamma_{\text{proj}}(m)=\pi(m-1)m^{m/2}$ [13, Theorem 1.3], rather than the state-of-the-art Steiner point method which can achieve $\gamma_{\text{Steiner}}(m)=m/2$ [46]. As our case studies show, in practice the projection-based NCBC algorithm performs much better than the worst-case bound. We note that any other NCBC algorithm with a finite competitive ratio can be used in (6.8a) in Algorithm 6. Investigating whether widely-used estimation methods, like least squares, have a finite competitive ratio would be an interesting avenue for future research.

Note that for Theorem 7 to hold, the optimization problem for the robust control oracle Π should first be solved without the slack variable ξ in Algorithm 6. This ensures that if $(\hat{X}_t, \hat{\eta}_t)$ is sufficiently close enough to the true model, then the algorithm will not make a mistake. In the case that Π is infeasible initially (*e.g.*, when the initial model estimate is far from the true model), it should be solved again with a slack variable, which ensures feasibility. However, solving Π twice is unnecessary in practice, and so we have written Algorithm 6 to reflect its practical implementation.

We outline a proof of Theorem 7 in the next section. We want to highlight one piece of that proof that is of independent interest. In particular, a major step in the proof is to provide a feasibility guarantee for the robust control oracle component Π of the algorithm, which is done in Theorem 9.

6.4 Proofs

We now prove our main result Theorem 7. Our proof builds on and adapts the approach of [111], which outlines a general framework for integrating model chasing and robust control. To explain the general framework, we first consider a discrete-time nonlinear dynamical system

$$x_{t+1} = f_*(x_t, u_t) + w_t, \qquad x_0 \text{ given}, \qquad (f_*, \mathbf{w}) \in \mathcal{F},$$

where $x \in \mathcal{S} \subseteq \mathbb{R}^n$ is the system state and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input. The unknown function f_* and disturbance sequence $\mathbf{w} \in \ell^\infty(\mathbb{Z}_+; \mathbb{R}^n)$ belong to an uncertainty set $\mathcal{F}$, and the disturbance is bounded as $\|\mathbf{w}\|_\infty \leq \overline{\eta}$. Assume that $\mathcal{F}$ has a *compact parametrization* $(\mathbb{T}, \mathsf{K}, d)$, where $\mathbb{T} : \mathsf{K} \to \wp(\mathcal{F})$ is a mapping from a parameter space K to a set of functions and disturbances such that $\mathcal{F} \subseteq \bigcup_{\theta \in \mathsf{K}} \mathbb{T}[\theta]$. $\wp(\mathcal{F})$ denotes the powerset of $\mathcal{F}$. Let d denote a metric on K, so (K, d) is a compact metric space.

The control objective is specified as a sequence of indicator "goal" functions $\mathcal{G} = (\mathcal{G}_0, \mathcal{G}_1, \dots)$. Each $\mathcal{G}_t : \mathcal{X} \times \mathcal{U} \to \{0, 1\}$ encodes a desired condition per time step t:

$$\mathcal{G}_t(x_t, u_t) = \mathbf{1}[x_t, u_t \text{ violate desired condition at time } t].$$

The main result of [111] specifies a set of sufficient conditions for a finite-mistake guarantee—*i.e.*, $\sum_{t=0}^{\infty} \mathcal{G}_t(x_t, u_t) < \infty$. These conditions decouple online robust control into separate online learning and robust control components. The online learning component requires a consistent model chasing algorithm SEL, which takes as input the current observed trajectory $D_t = [(x_i, x_{i+1}, u_i)]_{i=1}^t$ and outputs an estimated parameter $\theta_t \in \mathsf{K}$ which must be *consistent* with D_t.

Definition 6 (Consistent Parameter)**.** *We say $\theta \in \mathsf{K}$ is consistent with D_t if there exists $(f, \mathbf{w}) \in \mathbb{T}[\theta]$ such that*

$$\forall (x_t, x_{t+1}, u_t) \in D_t : \quad x_{t+1} = f(x_t, u_t) + w_t.$$

Let P_t denote the set of all parameters consistent with D_t; P_t is called the *consistent set*. We say SEL is γ-competitive if $\sum_{t=1}^{\infty} d(\theta_t, \theta_{t-1}) \leq \gamma \max_{\theta \in \mathsf{K}} d(P_\infty, \theta)$ holds for a fixed constant $\gamma > 0$, which we call the *competitive ratio*.

The robust control component requires a control oracle Π, which given the current state x_t and a parameter θ_t, outputs a control action $u_t = \Pi_{\theta_t}(x_t)$ that is robust for all systems that are close to θ_t. In particular, we call a control oracle ρ-*robust* for control objective $\mathcal{G}$, if all trajectories in $S^{\Pi}[\rho; \theta]$ achieve $\mathcal{G}$ after finitely many mistakes. $S^{\Pi}[\rho; \theta]$ is defined as the set of all possible trajectories generated by $\Pi_{\hat{\theta}}$ for all $\hat{\theta}$ such that $d(\theta, \hat{\theta}) \leq \rho$:

$$S^{\Pi}[\rho; \theta] = \left\{ \begin{array}{l|l} D_\infty = [(x_t, x_{t+1}, u_t)]_{t=1}^{\infty} : & \\ \quad u_t = \Pi_{\hat{\theta}}(x_t) & (f, \mathbf{w}) \in \mathbb{T}[\theta], \\ \quad x_{t+1} = f(x_t, u_t) + w_t & d(\hat{\theta}, \theta) \leq \rho \end{array} \right\}$$

We refer readers to [111] for a more detailed discussion of consistent model chasing algorithms and ρ-robust control oracles. As a summary, if SEL chases consistent models and Π is a robust oracle for $\mathcal{G}$, then the resulting $A_{\Pi}(\mathsf{SEL})$ algorithm achieves a finite mistake guarantee, which is stated in the following.

Theorem 8. *[111, Theorem 2.5] Assume that SEL chases consistent models and Π is a robust oracle for objective $\mathcal{G}$. Then for any starting point x_0 and trajectory $[(x_t, u_t)]_{t=0}^{\infty}$ generated by $\mathcal{A}_{\Pi}(\mathsf{SEL})$ (illustrated in Figure 6.1), the following mistake guarantees hold: (i) If Π is robust, then $\sum_{t=0}^{\infty} \mathcal{G}_t(x_t, u_t) < \infty$; (ii) If Π is uniformly ρ-robust and SEL is γ-competitive, then*

$$\sum_{t=0}^{\infty} \mathcal{G}_t(x_t, u_t) < \max\left\{1, M_\rho^{\Pi}\right\} \left(\frac{2\gamma}{\rho} \operatorname{diam}(\mathsf{K}) + 1 \right)$$

where M_ρ^{Π} denotes the worst case total mistakes of the ρ-robust control oracle Π.

To apply Theorem 8 to prove Theorem 7, we need to prove that (i) the proposed algorithm (6.8) chases consistent models and has a bounded competitive ratio, and (ii) the proposed robust algorithm in (6.11) is a ρ-robust control oracle, for

bounded disturbance in the system topology. In particular, the correspondence of the definitions is as follows. We have $\theta = (X, \eta)$, and

$$
\begin{aligned}
&\mathsf{K} = \mathcal{X} \times [0, \overline{\eta}], \quad v(1), q^c(0) \text{ given} \\
&d((X,\eta),(X',\eta')) = \|(X,\eta) - (X',\eta')\|_{\triangle,\delta} \\
&\mathbb{T}[(X,\eta)] = \left\{ (f, \mathbf{w}) \;\middle|\; \begin{array}{l} f(v,u) = v + Xu,\ \|\mathbf{w}\|_\infty \le \eta, \\ \forall t \ge 0 :\ \widehat{v_0^{\text{par}}} + \sum_{\tau=1}^{t} w(\tau) \in \mathcal{V}^{\text{par}} \\ \text{where } \widehat{v_0^{\text{par}}} := v(1) - Xq^c(0) \end{array} \right\} \\
&\mathcal{F} = \bigcup_{(X,\eta)\in\mathcal{X}\times[0,\overline{\eta}]} \mathbb{T}[(X,\eta)] \\
&\mathcal{G}_t(v(t)) = \mathbf{1}[v(t) \in [\underline{v}, \overline{v}]].
\end{aligned}
$$

We begin by proving that the set P_t defined in (6.8b) in Algorithm 6 is consistent with the trajectory D_t.

Lemma 11 (SEL is consistent)**.** *Suppose D_T is a trajectory of voltage measurements and control actions taken up to time T:*

$$D_T = [(v(t), v(t+1), u(t), q^c(t))]_{t=1}^{T}.$$

The set

$$
P_T := \left\{ (\hat{X}, \hat{\eta}) \;\middle|\; \begin{array}{l} \hat{X} \in \mathcal{X},\ \hat{\eta} \in [0, \overline{\eta}], \\ \forall (v(t), v(t+1), u(t), q^c(t)) \in D_T : \\ \quad \|v(t+1) - v(t) - \hat{X}u(t)\|_\infty \le \hat{\eta} \\ \quad v(t+1) - \hat{X}q^c(t) \in \mathcal{V}^{\text{par}} \end{array} \right\} \tag{6.10}
$$

is a consistent set for D_T, i.e., $(\hat{X}, \hat{\eta})$ is consistent (Definition 6) if and only if $(\hat{X}, \hat{\eta}) \in P_T$.

Proof. Consider any $(\hat{X}, \hat{\eta}) \in P_T$. For $t \in [T]$, define

$$\hat{f}(v,u) := v + \hat{X}u, \qquad \hat{w}(t) := v(t+1) - v(t) - \hat{X}u(t)$$

so $\|\hat{w}(t)\|_\infty \le \hat{\eta}$ and $v(t+1) = \hat{f}(v(t), u(t)) + \hat{w}(t)$. Define $\widehat{v_0^{\text{par}}} := v(1) - \hat{X}q^c(0)$, so for all $t \ge 0$,

$$\widehat{v_0^{\text{par}}} + \sum_{\tau=1}^{t} w(\tau) = v(t+1) - \hat{X}q^c(t) \in \mathcal{V}^{\text{par}}.$$

Thus, $(\hat{f}, \hat{w}) \in \mathbb{T}[(\hat{X}, \hat{\eta})]$, so $(\hat{X}, \hat{\eta})$ is consistent with D_T.

Conversely, suppose $(\hat{X}, \hat{\eta})$ is consistent with D_T, which implies the existence of $\hat{f}(v, u) := v + \hat{X}u$ and $\hat{\mathbf{w}}$ satisfying $\|\hat{\mathbf{w}}\|_\infty \le \hat{\eta}$ such that $v(t+1) = \hat{f}(v(t), u(t)) + \hat{w}(t)$. Rearranging yields $\hat{w}(t) = v(t+1) - v(t) - \hat{X}u(t)$, so $(\hat{X}, \hat{\eta})$ satisfies the norm constraint in (6.10). Now define

$$\forall t \ge 0: \quad \widehat{v^{\mathrm{par}}}(t) := v(t+1) - \hat{X}q^c(t) = \widehat{v_0^{\mathrm{par}}} + \sum_{\tau=1}^{t} \hat{w}(\tau)$$

so $\widehat{v^{\mathrm{par}}}(t) \in \mathcal{V}^{\mathrm{par}}$ satisfies the remaining constraint in (6.10). □

Observe that each P_t is a closed, bounded, and convex set. Furthermore, P_t is non-empty, since $(X^\star, \eta^\star) \in P_t$. Intuitively, P_t is the smallest set containing all parameters that could generate the observed trajectory D_t along with a corresponding admissible sequence of noise compatible with Assumptions 9 to 11.

The consistent sets are nested $P_t \subseteq P_{t-1}$, and each P_t is a nonempty compact convex subset of the parameter space $\mathcal{X} \times [0, \overline{\eta}]$. To apply the Euclidean NCBC results of [13, 46], we use the linear map

$$\Phi(X, \eta) := [\mathrm{vech}(X),\ \delta\eta]^\top,$$

which is an isometric identification between $(\mathcal{X} \times [0, \overline{\eta}], \|\cdot\|_{\triangle,\delta})$ and a subset of Euclidean space $(\mathbb{R}^m, \|\cdot\|_2)$. This reduces the model-chasing component to standard NCBC in $m = \frac{n(n+1)}{2} + 1$ dimensions. The corresponding lemma is as follows.

Lemma 12 (SEL is competitive)**.** *Suppose the NCBC algorithm applied by* SEL *is $\gamma(m)$-competitive on nested convex subsets of Euclidean space $\mathbb{R}^m$. Then, under the identification $\Phi(X, \eta) = [\mathrm{vech}(X),\ \delta\eta]^\top$, the model-chasing algorithm* SEL *is $\gamma(m)$-competitive on $(\mathcal{X} \times [0, \overline{\eta}], \|\cdot\|_{\triangle,\delta})$.*

Proof. Let $m = \frac{n(n+1)}{2} + 1$. The normed vector space $(\mathbb{S}^n \times \mathbb{R}, \|\cdot\|_{\triangle,\delta})$ is isometrically isomorphic to the Euclidean space $(\mathbb{R}^m, \|\cdot\|_2)$ because the map $\Phi(X, \eta) = [\mathrm{vech}(X),\ \delta\eta]^\top$ is a linear bijection from $\mathbb{S}^n \times \mathbb{R}$ to $\mathbb{R}^m$ and preserves the norm:

$$\|\Phi(X, \eta) - \Phi(X', \eta')\|_2^2 = \|X - X'\|_\triangle^2 + \delta^2|\eta - \eta'|^2 = \|(X, \eta) - (X', \eta')\|_{\triangle,\delta}^2.$$

Therefore, the NCBC competitive guarantee transfers directly from Euclidean space to the parameter space used by SEL, so any $\gamma(m)$-competitive NCBC algorithm in $\mathbb{R}^m$ induces a $\gamma(m)$-competitive selector in our setting.

When $\eta^\star$ is known, the map $\mathrm{vech}(\cdot)$ is an isometric identification between $(\mathbb{S}^n, \|\cdot\|_\triangle)$ and Euclidean space $(\mathbb{R}^{n(n+1)/2}, \|\cdot\|_2)$, so the same argument applies with $m = \frac{n(n+1)}{2}$. □

In our experiments, we instantiate SEL with the greedy projection algorithm from [13]. Applying [13, Theorem 1.3] and noting that $\forall d \in \mathbb{N} : \ (\omega_d/\omega_{d-1}) \leq \pi$, where ω_d denotes the surface area of the d-sphere, we obtain the explicit Euclidean bound $\gamma(m) = \pi(m-1)m^{m/2}$. This concrete bound is useful for interpreting the worst-case mistake guarantee in Theorem 7, but it is not needed for the abstract reduction above.

Finally, we show that our controller Π is ρ-robust. In particular, we prove that $\Pi_{\hat{X}}$ makes no mistakes ($M_\rho^\Pi = 0$) given consistent parameters $(\hat{X}, \hat{\eta}) \in P_t$.

Theorem 9 (Π is ρ-robust)**.** *Under Assumptions 9 to 11, suppose $(\hat{X}, \hat{\eta}) \in P_t$, where P_t is given in (6.10) for $t \geq 1$. Define $\rho = \frac{\delta\epsilon}{1+\delta\|\overline{q}-\underline{q}\|_2}$. Then, the following optimization problem is feasible:*

$$\min_{u \in \mathbb{R}^n} \quad \|\hat{v}' - v^{\mathrm{nom}}\|_{P_v}^2 + \|u\|_{P_u}^2 \tag{6.11a}$$

$$s.t. \quad \underline{q} \preceq q^c(t-1) + u \preceq \overline{q} \tag{6.11b}$$

$$\hat{v}' = v(t) + \hat{X}u \tag{6.11c}$$

$$k = \hat{\eta} + \rho\left(\frac{1}{\delta} + \|u\|_2\right) \tag{6.11d}$$

$$\underline{v} + k\mathbf{1} \preceq \hat{v}' \preceq \overline{v} - k\mathbf{1}. \tag{6.11e}$$

Further, the solution of (6.11), $u(t)$, guarantees voltage stability for all $(X, \eta) \in \mathcal{X} \times [0, \overline{\eta}]$ such that $\|(X, \eta) - (\hat{X}, \hat{\eta})\|_{\triangle,\delta} \leq \rho$. That is, $v(t) + Xu(t) + w(t) \in [\underline{v}, \overline{v}]$ for all $w(t)$ such that $\|w(t)\|_\infty \leq \eta$.

Observe that (6.11) corresponds to (6.9) in Algorithm 6 with the slack variable set to zero. We note that the robustness margin ρ decreases as $[\underline{q}, \overline{q}]$ increase. The intuitive reason is that the voltage is more sensitive to changes in $\hat{X}$ when the range of possible u's expands. Therefore, a fixed voltage buffer of ϵ in constraints (6.9d) and (6.11d) affords less robustness to changes in $\hat{X}$ as $[\underline{q}, \overline{q}]$ gets larger.

Proof of Theorem 9. First, we will show that the following two conditions are sufficient for feasibility of the optimization problem and ρ-robustness for the solution u.

- Feasibility: $k \le \overline{\eta} + \epsilon$
- Robustness: $k \ge \hat{\eta} + \rho\sqrt{\frac{1}{\delta^2} + \|u\|_2^2}$

Then, we will show that our choices of k and ρ satisfy these sufficient conditions.

To derive the sufficient condition for feasibility, define

$$\widehat{v^{\text{par}}}(t-1) := v(t) - \hat{X}q^c(t-1)$$

as the conjectured noise when we assume the underlying parameter is $\hat{X}$. Since $\hat{X} \in P_t$ and $P_t \subseteq P_{t-1}$, we have $\widehat{v^{\text{par}}}(t-1) \in \mathcal{V}^{\text{par}}$. Then, by Assumption 11, there exists $q^c \in [\underline{q}, \overline{q}]$ such that

$$\underline{v} + (\overline{\eta} + \epsilon)\mathbf{1} \preceq \hat{X}q^c + \widehat{v^{\text{par}}}(t-1) \preceq \overline{v} - (\overline{\eta} + \epsilon)\mathbf{1}.$$

Set $u = q^c - q^c(t-1)$ (which satisfies (6.11b)) and define

$$\begin{aligned}\hat{v}'(u) &:= v(t) + \hat{X}u = v(t) + \hat{X}[q^c - q^c(t-1)] \\ &= \hat{X}q^c + \widehat{v^{\text{par}}}(t-1).\end{aligned}$$

Recalling (6.6), we can interpret $\hat{v}'(u)$ as the one-step voltage prediction (without disturbance) under the model $\hat{X}$ given control action u and the current voltage $v(t)$. We thus have

$$\underline{v} + (\overline{\eta} + \epsilon)\mathbf{1} \preceq \hat{v}'(u) \preceq \overline{v} - (\overline{\eta} + \epsilon)\mathbf{1}.$$

Therefore, as long as $k \le \overline{\eta} + \epsilon$, u will satisfy constraint (6.11e).

Next, we derive the sufficient condition for robustness. Let u be a solution of (6.11), so it satisfies (6.11e). Let $(X, \eta) \in \mathcal{X} \times [0, \overline{\eta}]$ be arbitrary parameters satisfying $\|(X, \eta) - (\hat{X}, \hat{\eta})\|_{\triangle,\delta} \le \rho$. Define $\rho_X := \|X - \hat{X}\|_{\triangle}$. By Lemma 13,

$$-\rho_X \|u\|_2 \mathbf{1} \preceq (X - \hat{X})u \preceq \rho_X \|u\|_2 \mathbf{1}. \tag{6.12}$$

Furthermore, suppose

$$-\eta\mathbf{1} \preceq w(t) \preceq \eta\mathbf{1}. \tag{6.13}$$

Adding together the 3 inequalities (6.11e), (6.12), (6.13) yields

$$\begin{aligned}\underline{v} + (k - \rho_X \|u\|_2 - \eta)\mathbf{1} &\preceq v(t) + Xu + w(t) \\ &\preceq \overline{v} - (k - \rho_X \|u\|_2 - \eta)\mathbf{1}.\end{aligned}$$

Clearly, if $k - \rho_X \|u\|_2 - \eta \ge 0$, then the desired robustness condition is satisfied. Since

$$\|(X, \eta) - (\hat{X}, \hat{\eta})\|_{\triangle,\delta}^2 = \rho_X^2 + \delta^2 |\eta - \hat{\eta}|^2 \le \rho^2,$$

we have $|\eta - \hat{\eta}| \leq \frac{1}{\delta}\sqrt{\rho^2 - \rho_X^2}$. This implies $\eta \leq \hat{\eta} + \frac{1}{\delta}\sqrt{\rho^2 - \rho_X^2}$. Therefore, we can express the robustness condition in terms of $\hat{\eta}$:

$$k \geq \hat{\eta} + \frac{1}{\delta}\sqrt{\rho^2 - \rho_X^2} + \rho_X \|u\|_2 =: f(\rho_X).$$

For $\rho > 0$, $f(\rho_X)$ is strictly concave and twice-differentiable and therefore achieves its maximum when $f'(\rho_X) = 0$. This maximum value is $\hat{\eta} + \rho\sqrt{\frac{1}{\delta^2} + \|u\|_2^2}$. Thus, if k is at least this value, then we achieve robustness.

Finally, we show that our choices of k and ρ satisfy the sufficient conditions. Since $a + b \geq \sqrt{a^2 + b^2}$ for all $a, b \geq 0$, our choice of k satisfies the robustness condition:

$$k = \hat{\eta} + \rho\left(\frac{1}{\delta} + \|u\|_2\right) \geq \hat{\eta} + \rho\sqrt{\frac{1}{\delta^2} + \|u\|_2^2}.$$

Note that while setting $k = \hat{\eta} + \rho\sqrt{\frac{1}{\delta^2} + \|u\|_2^2}$ would also satisfy the robustness condition, this expression would make (6.11) a nonconvex optimization problem.

The remaining step is to satisfy the feasibility condition. We must choose ρ such that $\hat{\eta} + \rho\left(\frac{1}{\delta} + \|u\|_2\right) \leq \overline{\eta} + \epsilon$. Since $\hat{\eta} \leq \overline{\eta}$, it suffices to find ρ such that $\rho\left(\frac{1}{\delta} + \|u\|_2\right) \leq \epsilon$. As $\|u\|_2 \leq \|\overline{q} - \underline{q}\|_2$, setting $\rho = \frac{\delta\epsilon}{1 + \delta\|\overline{q} - \underline{q}\|_2}$ satisfies the inequality. □

In the case where $\eta^\star$ is known, a similar proof shows that $k = \eta^\star + \rho\|u\|_2$ and $\rho = \frac{\epsilon}{\|\overline{q} - \underline{q}\|_2}$ satisfy feasibility and robustness. (This can be seen as the $\delta \to \infty$ limiting case of Theorem 9 such that consistent model chasing only updates $\hat{X}$ and keeps $\hat{\eta} = \eta^\star$ fixed.)

Lemma 13. *For all $A \in \mathbb{S}^n$, $b \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}_+$,*

$$\|A\|_\triangle \leq \alpha \quad \Longrightarrow \quad -\alpha\|b\|_2 \mathbf{1} \preceq Ab \preceq \alpha\|b\|_2 \mathbf{1}.$$

Proof. Let A_i denote the i^{th} row of A. By symmetry of A,

$$\begin{aligned}\|A_i\|_2^2 &= \sum_{j=1}^n A_{i,j}^2 = \sum_{k=1}^{i-1} A_{k,i}^2 + \sum_{j=i}^n A_{i,j}^2 \\ &\leq \sum_{k=1}^n \sum_{j=k}^n A_{k,j}^2 = \|A\|_\triangle^2 \leq \alpha^2,\end{aligned}$$

so $\|A_i\|_2 \leq \alpha$. Then

$$-\alpha\|b\|_2 \leq -\|A_i\|_2\|b\|_2 \leq (Ab)_i \leq \|A_i\|_2\|b\|_2 \leq \alpha\|b\|_2.$$

This holds for all $i \in \{1, \ldots, n\}$, which yields the desired result. □

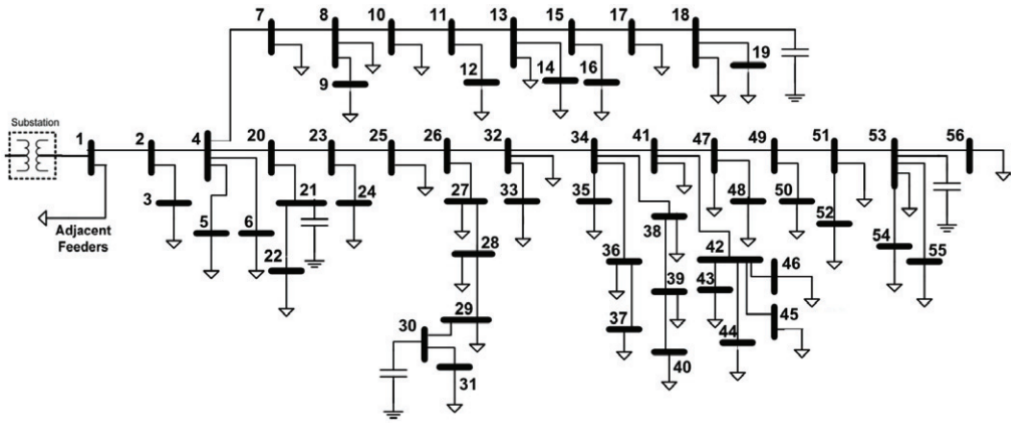


Figure 6.2: Schematic diagram of SCE 56 bus distribution system.

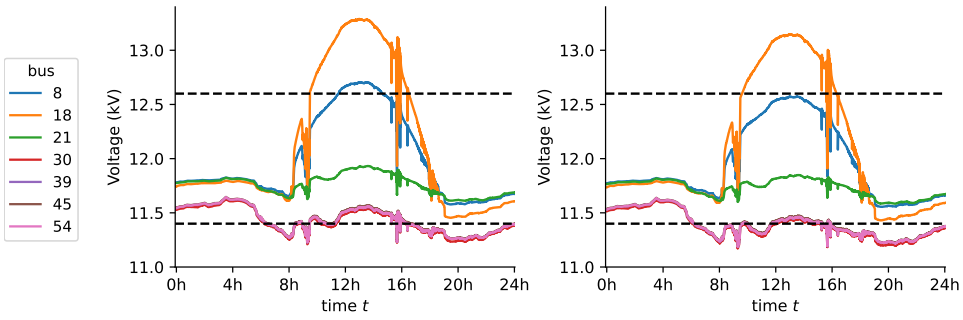


Figure 6.3: Voltage profile of 7 buses without control, simulated with **(a)** linear dynamics (6.2) and **(b)** nonlinear balanced AC dynamics (6.1).

Finally, combining Theorem 9 with Lemma 12 and applying Theorem 8 completes the proof of Theorem 7.

6.5 Case Study

We demonstrate the effectiveness of Algorithm 6 using a case study based on a single-phase 56-bus network ($n = 55$) from the Southern California Edison (SCE) utility (Figure 6.2), with line parameters r_{ij}, x_{ij} from [84, Table 1]. Even though our algorithm only has guarantees for the linear power flow model (6.2), we show that our algorithm works well on both the linear model and the more realistic nonlinear DistFlow model (6.1).

Experimental Setup

Following [210], we adapt real-world load and PV data from [27] for the 56-bus network by adding power injection (scaled by the PV generation) at buses $C = \{2, 4, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 19, 20, 23, 25, 26, 32\}$. Exogenous active

and reactive power injection measurements are taken at each bus at 6-second intervals over a 24-hour period. Figure 6.3 plots these values for several buses to illustrate the setting considered. We assume that controllers with reactive power injection capacity are available at every node. The network parameters used in our experiments are:

- nominal squared voltage magnitude at the substation $v^0 = v^{\text{nom}} = (12\text{kV})^2$
- squared voltage magnitude limits $[\underline{v}, \overline{v}] = [0.95, 1.05]\text{pu} = [11.4^2, 12.6^2]\text{kV}^2$
- reactive power injection limits $[\underline{q}, \overline{q}] = [-0.24, 0.24]\text{MVar}$
- state and input cost matrices $P_v = 0.1I$, $P_u = 10I$
- initial state $v(1) = R^\star p(0) + X^\star q^e(0) + v^0\mathbf{1}$, $q^c(0) = \mathbf{0}$

In comparison to previous papers in the voltage control literature, our reactive power injection limits $[\underline{q}, \overline{q}]$ are slightly more generous than ± 0.2 MVar used in, *e.g.*, [210]. We choose ± 0.24 MVar because even a controller with perfect knowledge of the future would need reactive power injection capabilities of at least ± 0.238 MVar in order to maintain $v(t) \in [\underline{v}, \overline{v}]$ (if $\underline{q} = -\overline{q}$) under linear dynamics (6.2).

We set $\overline{\eta} = 10$, which upper-bounds the maximum change in exogenous noise observed in our data, which is ≈ 8.6:

$$\eta^\star = \max_t \left\| R^\star(p(t) - p(t-1)) + X^\star(q^e(t) - q^e(t-1)) \right\|_\infty .$$

We fix $\epsilon = 0.1$. In order to satisfy the requirement in Assumption 11 that $v(t) \in [\underline{v} + (\overline{\eta} + \epsilon), \overline{v} - (\overline{\eta} + \epsilon)]$, the reactive power injection capabilities must exceed ± 0.528 MVar. As we show in experiments with only ± 0.24 MVar range of control, though, Assumption 11 does not need to be fully satisfied in order for our method to still provide strong empirical results.

For the robust controller Π, we set slack variable weight $\beta = 100$ and $\mathcal{V}^{\text{par}} = [\underline{v^{\text{par}}}, \overline{v^{\text{par}}}]$ to be a rectangle around the true noise. Under linearized system dynamics, $v^{\text{par}}(t)$ is calculated as described in Section 6.2, and then we set

$$\forall i \in [n] : \quad \underline{v^{\text{par}}}_i = \min_t v_i^{\text{par}}(t), \quad \overline{v^{\text{par}}}_i = \max_t v_i^{\text{par}}(t).$$

Under nonlinear system dynamics, we approximate $v^{\text{par}}(t)$ as the nodal squared voltage magnitudes when $q^c(t) = 0$ (as shown in Figure 6.3), and we add 0.5kV^2 padding which empirically suffices as a convex outer approximation of $\mathcal{V}^{\text{par}}$:

$$\underline{v^{\text{par}}}_i = \min_t v_i^{\text{par}}(t) - 0.5, \qquad \overline{v^{\text{par}}}_i = \max_t v_i^{\text{par}}(t) + 0.5.$$

As mentioned previously, we use a greedy projection-based NCBC algorithm [13] in SEL that minimizes the movement distance $\|(\hat{X}_t, \hat{\eta}_t) - (\hat{X}_{t-1}, \hat{\eta}_{t-1})\|_{\triangle,\delta}$ between

Table 6.1: Performance of our method simulated under linear system dynamics (top) and nonlinear system dynamics (bottom). See Section 6.5.

Info provided	# mistakes	avg. violation	max violation
Unknown	662.2 ± 435.1	0.43 ± 0.16	4.40 ± 2.59
Topo-14	917.0 ± 155.2	0.34 ± 0.12	4.93 ± 2.19
Lines-14	1085.8 ± 186.6	0.57 ± 0.29	2.55 ± 1.09
Known	88.0	0.07	0.12
Unknown	16.0 ± 15.8	0.68 ± 0.56	2.74 ± 2.39
Topo-14	0.5 ± 0.6	2.21 ± 2.56	2.90 ± 3.38
Lines-14	0.5 ± 0.6	1.01 ± 1.20	1.45 ± 1.73
Known	0.0	0.00	0.00

nested convex sets $P_t \subseteq P_{t-1}$:

$$\mathrm{NCBC}_{\mathrm{proj}}(P_t, \hat{X}_{t-1}, \hat{\eta}_{t-1}) := \operatorname*{arg\,min}_{(X,\eta)\in P_t} \|(X,\eta) - (\hat{X}_{t-1}, \hat{\eta}_{t-1})\|_{\triangle,\delta}. \tag{6.14}$$

This achieves competitive ratio $\gamma_{\mathrm{proj}}(m) = \pi(m-1)m^{m/2}$.

To keep the optimization problem (6.8) computationally tractable for consistent model chasing, our implementation does not use the full trajectory D as in the constraints of the consistent set (6.10). Instead, we include the 20 latest observations and 80 more observations sampled uniformly at random $(v(t), v(t+1), u(t), q^c(t)) \sim D$. This provides a computationally tractable approximation of the uncertainty set. In our experiments on linear system dynamics, we found that $\hat{X}_t$ selected using this approximation was always in the consistent set defined by the full trajectory D, when allowing for small numerical inaccuracies introduced by the CVXPY optimization solver.

Unless otherwise stated, we initialize $\hat{\eta}_1 = 0$. We initialize $\hat{X}_1$ by adding noise to the true $X^\star$ in two ways. First, we scale each line impedance x_{ij} by a random factor $\sigma_{ij} \overset{\text{iid}}{\sim} \mathrm{Uniform}[0,2]$. Second, we randomly permute the bus ordering, so $\hat{X}_1$ corresponds to a permuted grid topology. Finally, we project $\hat{X}_1$ into the uncertainty set $\mathcal{X}_\alpha$, with $\alpha = 1$.

Except for the experiments shown in Figure 6.6, we fix $\delta = 20$ which empirically strikes a balance between minimizing the modeling error $\|\hat{X}_t - X^\star\|_\triangle$ and overfitting noise.

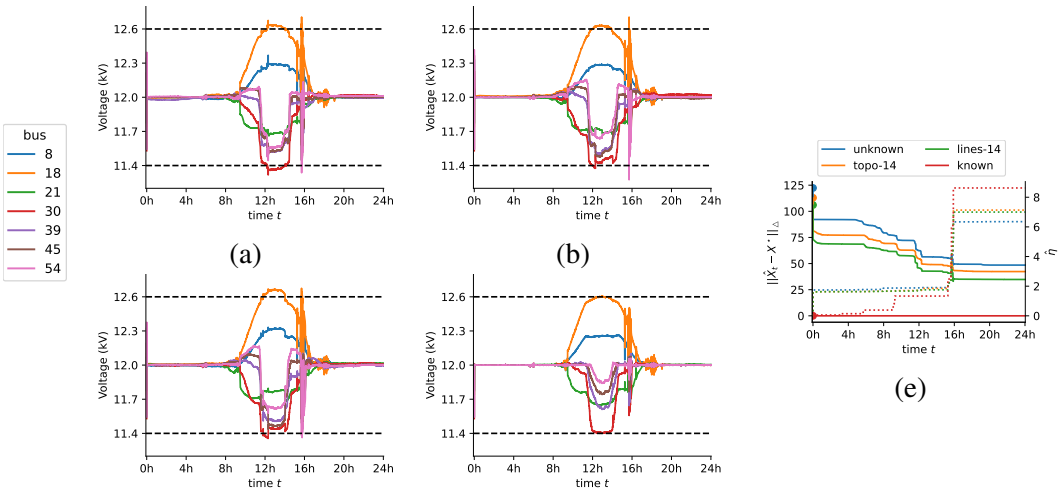


Figure 6.4: **(a)-(d)** Voltage profiles of 7 different buses simulated under linear system dynamics (6.2). Dotted black lines indicate voltage limits $[\underline{v}, \overline{v}]$. **(a)** Π+SEL initialized with random $\hat{X} \in \mathcal{X}_\alpha$. **(b)** like (a) but the topology for buses 1-14 is known. **(c)** like (a) but the topology and line parameters for buses 1-14 are known. **(d)** like (a) but $\hat{X} = X^\star$ is fixed and known so only $\hat{\eta}$ is learned **(e)** Convergence of $\hat{X}_t$ towards true $X^\star$ (solid lines, left axis) and estimated $\hat{\eta}$ (dotted lines, right axis). Notice that even when $\|\hat{X}_t - X^\star\|_\triangle$ does not reach 0, the controller still performs quite well.

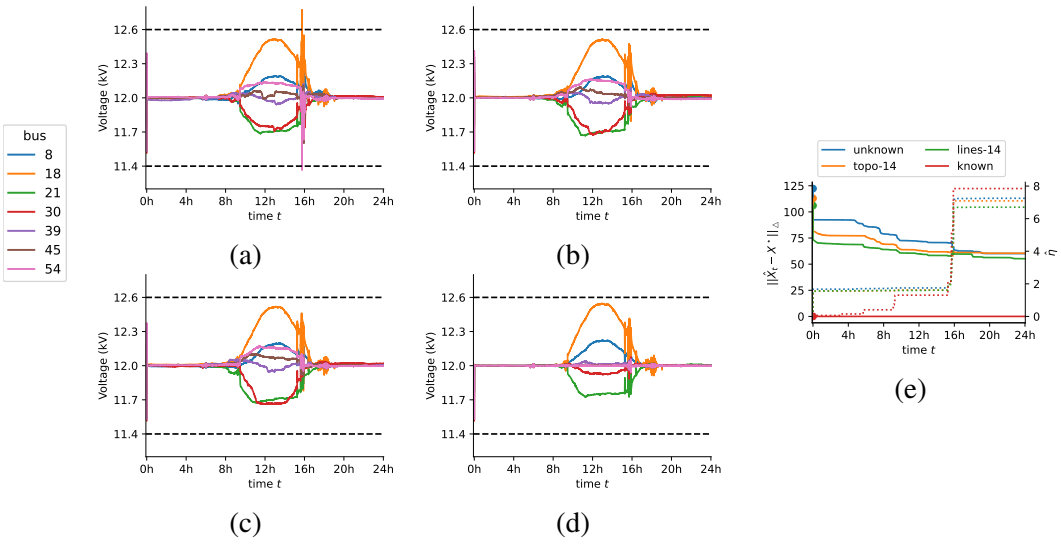
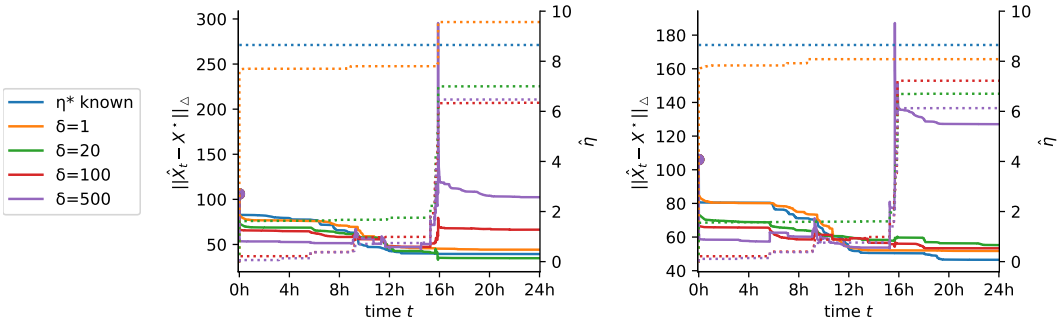
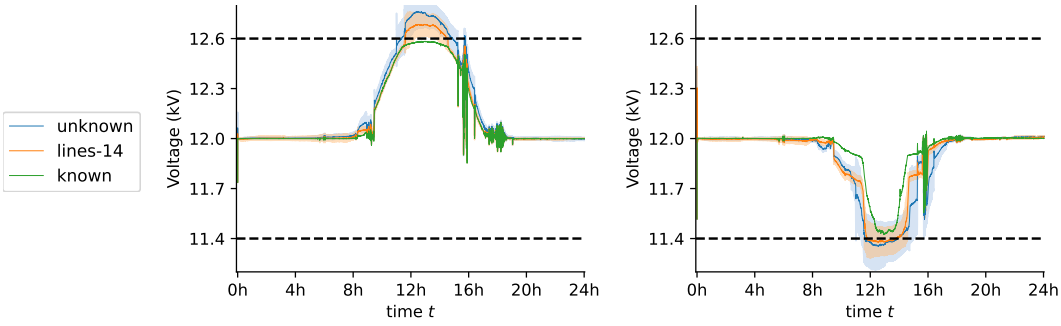


Figure 6.5: Voltage profiles of 7 different buses simulated under balanced nonlinear AC power flow (6.1). Parallels Figure 6.4.



(a) (b)

Figure 6.6: Effect of varying δ on consistent model chasing. As in Figure 6.4e, convergence of $\hat{X}_t$ towards $X^\star$ is plotted in solid lines (left axis), and estimated $\hat{\eta}$ is plotted in dotted lines (right axis). In blue are results where we fix $\hat{\eta} = \eta^\star = 8.65$ and δ has no effect. **(a)** linear dynamics **(b)** nonlinear dynamics.



(a) (b)

Figure 6.7: Balanced nonlinear AC power flow simulation of the voltage profiles under different algorithms with *partial control and observation*. The dark colors plot the mean voltages across 4 random initializations of $\hat{X}_1$ and the light shading plots ± 1 standard deviation. **(a)** bus 18 **(b)** bus 30.

Experimental Results

Our experimental results demonstrate the ability of Algorithm 6 to stabilize the system without knowledge of the network topology, providing good voltage control performance even though it still has significant uncertainty about the topology at the end of the experiments. We test our algorithm under both the linearized system dynamics (6.5) as well as the more realistic nonlinear balanced AC power flow setting (6.1) simulated using Pandapower [269]. The convex optimization problems for SEL and Π are solved with CVXPY [70] using the MOSEK solver [182]. Code for our simulations is available on GitHub.[2]

[2] https://github.com/chrisyeh96/voltctrl

Linearized power flow with full control Our first set of experiments, shown in Figure 6.4 and Table 6.1 (top), tests our algorithm's performance on the SCE-56 bus network under linearized system dynamics (6.5). Different amounts of network information are provided to the consistent model chasing algorithm SEL via the initial consistent set $\mathcal{X}_\alpha$, ranging from no information ("unknown," Figure 6.4a), information about the edges among the first 14 buses but not the line impedances ("topo-14," Figure 6.4b), information about the edges *and* line impedances among the first 14 buses ("lines-14," Figure 6.4c), and complete information about the network ("known," Figure 6.4d). Because the buses in the SCE 56-bus network are numbered in a topological ordering, the "topo-14" setting adds constraints of the form (6.7) for all of the first 14 buses, and the "lines-14" setting constrains all $X \in \mathcal{X}_\alpha$ such that $X_{ij} = X^\star_{ij}$ for all $i, j \in \{1, \ldots, 14\}$.

As shown in Figure 6.4e, incorporating more prior knowledge about the network into the initial uncertainty set reduces the model estimation error $\|\hat{X} - X^\star\|_\triangle$. Furthermore, the model estimation error decreases the most dramatically when the voltage violations are the largest. However, we note that lower model estimation error does not always result in fewer mistakes in our experiments.

Table 6.1 quantifies our algorithm's performance under varying amounts of initial network information. A "mistake" refers to any time step where any bus' voltage violated the limits $[\underline{v}, \overline{v}]$. "Avg. violation" refers to the average absolute squared-voltage violation

$$\operatorname*{mean}_{i\in[n],\, t\in[T]:\ v_i(t)\notin[\underline{v}_i,\overline{v}_i]} \max(v_i(t) - \overline{v}_i,\ \underline{v}_i - v_i(t)).$$

"Max violation" is like "avg. violation" but replaces the mean with a max. Results given show the mean and standard deviation over 4 random initializations of $\hat{X}_1$.

Nonlinear power flow with full control Our second set of experiments test our online controller on the standard balanced AC power flow model (6.1). As in the linearized power flow experiments, we compare Algorithm 6's performance across varying levels of prior information (Figure 6.5 and Table 6.1, bottom). Even though the controller is designed under the assumption of linearized voltage dynamics, our algorithm still performs well in the nonlinear simulation. The performance improves progressively, with less voltage violation and smaller overall deviation from the desired steady state voltage as it is provided more information.

Nonlinear power flow with partial observation and partial control We also test our proposed online controller in the partial observation and partial control setting. In Figure 6.7, we withhold voltage observations and control authority from buses $i \in \{8, 18, 21, 30, 39, 45, 54\}$ by setting $q_i^c(t) = 0$ for all t. We simulate the voltage profiles across 4 random initializations of $\hat{X}_1$ and plot the mean and ± 1 standard deviation. Despite the more challenging setting, the performance of Algorithm 6 remains strong. We again observe in Figure 6.7 that adding prior topology and line parameter information marginally improves the performance of Algorithm 6.

Varying δ In Figure 6.6, we demonstrate the effect of varying δ on the performance of our algorithm. From a theoretical perspective, Theorem 7 shows that our algorithm achieves a finite mistake bound for every $\delta > 0$, and this bound is minimized by taking δ to be very large. What happens when using a large δ, though, is that the model chasing algorithm may overfit to noise until a time when the noise is too large, forcing the algorithm to increase the noise bound (*e.g.*, around the 16h mark in Figure 6.6). This leads to inconsistent performance in the short term, albeit with perhaps better worst-case performance. In contrast, a smaller δ allows more of the network uncertainty to be captured in a larger noise $\hat{\eta}$ term at the cost of learning a less accurate $\hat{X}$, but the decrease in modeling error $\|\hat{X}_t - X^\star\|_\triangle$ becomes monotonic.

In practice, δ should be treated as a prior "confidence" about how close the initial guess of $\hat{\eta}$ is to $\eta^\star$. δ should be larger when there is greater confidence that $\hat{\eta}$ is close to the true $\eta^\star$.

Detecting topology changes Finally, we consider the challenge of responding to a change in the distribution grid topology in real-time. If the topology changes from one radial grid to another due to switches, new observed data may render the consistent set empty. That is, when consistent model chasing (6.14) becomes infeasible, we are assured that the topology has changed. At this point, we may reset the algorithm by discarding the observed trajectory D_t and reinitializing consistent parameter estimates from the original consistent set P_1. Figure 6.8 demonstrates this on linear system dynamics, where we introduce a topology change at the 12h mark. We replace lines $33 \to 40$ and $46 \to 48$ with new lines $1 \to 40$ and $10 \to 48$, which maintains a radial distribution grid.

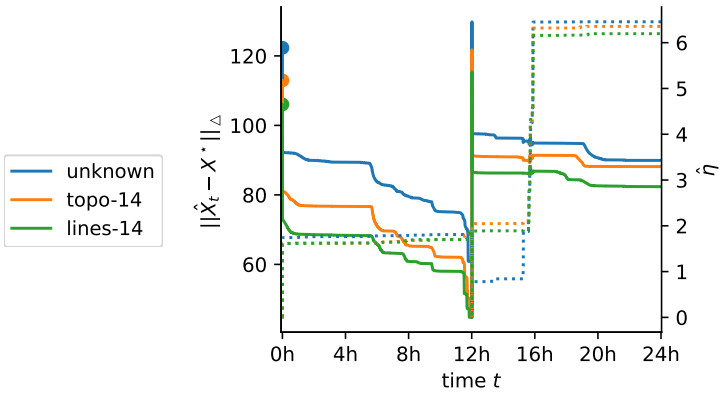


Figure 6.8: Demonstration of the detection of a topology change under linear system dynamics. Convergence of $\hat{X}_t$ towards $X^\star$ is plotted in solid lines (left axis), where $X^\star$ changes at the 12h mark. The topology change triggers a reset of the consistent model chasing algorithm. Estimated $\hat{\eta}$ is plotted in dotted lines (right axis).

6.6 Conclusion

This chapter provides the first controller that establishes a finite-mistake guarantee for voltage control in a setting with uncertainty in both the grid topology and load and generation variations. We showed that our proposed algorithm is able to learn a model of the grid dynamics in an online fashion and provably (under linearized voltage dynamics) converge to a stable controller. Further, simulated experiments on a 56-bus distribution grid demonstrate the effectiveness of our algorithm even under more realistic nonlinear dynamics. We demonstrated how to incorporate prior knowledge about the network topology and line parameters to improve performance, while also extending our algorithm to the partial observability and partial controllability setting which may better reflect real-world scenarios.

As the current algorithm is centralized, future works may consider decentralized approaches to topology-robust voltage control in order to enable faster real-time control with ideas from [315]. Another direction is to extend the current algorithm to the time-varying topology setting with techniques from works such as [314]. Further studies may also explore loosening the radial topology assumption and test our algorithm on unbalanced 3-phase AC grids to accommodate a wider range of distribution grids. This would be a challenging, but important, extension. Finally, an interesting algorithmic extension is to consider computationally efficient convex body chasing algorithms with better competitive ratios. Existing methods based on Steiner point [13, 46] achieve nearly-optimal competitive ratio but are computationally inefficient in high dimension settings such as voltage control, so designing efficient approximate Steiner point algorithms could potentially lead to significant performance

improvements.

Chapter 7

SUSTAINGYM: REINFORCEMENT LEARNING ENVIRONMENTS FOR SUSTAINABLE ENERGY SYSTEMS

In the second two chapters of this part of the thesis, we shift our focus to reinforcement learning (RL), which is a very general framework for sequential decision making under uncertainty. Unlike in previous chapters where we are able to leverage specific domain knowledge encoded in the structure of optimization problems, RL takes a more general approach to learning control policies directly from interactions with the environment. This makes RL a promising tool for optimizing complex systems, but it also presents unique challenges, particularly in the context of sustainability applications where data can be scarce and the consequences of suboptimal actions can be significant.

This chapter is focused on the development of SustainGym, a suite of RL environments designed to test the performance of RL algorithms on realistic sustainable energy system tasks. The environments cover a range of applications, including electric vehicle charging, carbon-aware data center job scheduling, and more. Each environment is designed to test RL algorithms under realistic distribution shifts and in multi-agent settings, which are common in real-world sustainability problems. We show that standard off-the-shelf RL algorithms leave significant room for improving performance and highlight the challenges ahead for introducing RL to real-world sustainability tasks. The next chapter will build upon the environments introduced in this chapter to investigate the problem of learning robust multi-agent policies that can perform well under distribution shifts, which is a critical challenge for deploying RL in real-world sustainability applications.

This chapter is primarily based on the following paper:

[310] Christopher Yeh et al. 2023. SustainGym: reinforcement learning environments for sustainable energy systems. In *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., New Orleans, LA, USA, (Dec. 2023), 59464–59476. https://proceedings.neurips.cc/paper_files/paper/2023/hash/ba74855789913e5ed36f87288af79e5b-Abstract-Datasets_and_Benchmarks.html.

7.1 Introduction

While reinforcement learning (RL) algorithms have demonstrated tremendous success in applications ranging from game-playing, *e.g.*, Atari and Go, to robotic control,

e.g., [178, 245, 137], most RL algorithms continue to only be benchmarked using toy environments—*e.g.*, OpenAI Gym [42]. These toy environments generally do not have realistic physical constraints, nor realistic environmental shifts over time. Furthermore, these environments are generally limited to single-agent systems, whereas real-world systems tend to involve coordination and/or competition between actors. The realism gap limits the reliable deployment of off-the-shelf RL algorithms in real-world systems.

Developing better RL algorithms to address these challenges requires a means of empirically benchmarking and comparing the performance of different algorithms in real-world settings. Our inspiration comes from progress in supervised machine learning (ML), where widespread adoption of breakthrough techniques was fueled by large datasets with standardized benchmarks, such as ImageNet for computer vision [225] and the GLUE benchmark for natural language processing [279]. More recently, many supervised learning datasets have been created to address specific real-world sustainability challenges, such as monitoring global progress towards sustainable development goals [309].

In this work, we introduce SustainGym, a suite of 5 RL environments that realistically model sustainability settings, summarized in Table 7.1:

- `EVChargingEnv` models the problem of scheduling electric vehicle (EV) charging to meet user needs while minimizing CO_2 emissions.
- `ElectricityMarketEnv` models a grid-scale battery storage system bidding into the electricity market to generate profit (through price arbitrage) and reduce CO_2 emissions.
- `DatacenterEnv` models a datacenter deciding on a "virtual capacity curve" to shift flexible jobs towards times of day with lower CO_2 emissions.
- `CogenEnv` models a combined cycle cogeneration plant producing steam and electricity to meet local demand while minimizing fuel usage and ramp costs.
- `BuildingEnv` models the thermal control of building energy systems to reduce the total electricity consumption while satisfying the user-specified temperature requirement.

A key feature of SustainGym environments is their support for testing RL algorithms under realistic and natural exogenous distribution shifts, which generally fall under two categories:

1. *Shifts in demand*. In each environment, RL agents choose actions to satisfy

Table 7.1: Summary of environments included in SustainGym and their features. The "Single agent" and "Multi-agent" rows indicate what an individual RL agent controls in that environment.

Env	EVChargingEnv	ElectricityMarketEnv	DatacenterEnv	CogenEnv	BuildingEnv
Control task	charging rates for EV charging stations	market bids for a grid-connected battery storage system	virtual capacity curve for a carbon-aware data center	dispatch set points for turbines	heating supply for buildings
Modeled after	charging networks at Caltech & JPL	generic test case (IEEE RTS-GMLC)	(loosely) a Google data center	specific combined cycle gas generation plant in the U.S.	generic DoE commercial reference building models
Single agent	all EV charging stations	single battery system	single data center	all 4 turbine units	all buildings
Multi-agent	one EV charging station, cooperative	N/A	N/A	one turbine unit, cooperative	one room, cooperative
Actions	discrete or continuous	discrete or continuous	continuous	mixed discrete & continuous	discrete or continuous
Rewards	cost + CO_2	cost + CO_2	penalty + CO_2	cost + CO_2	temperature difference + energy use
Distribution shift	MOER, EV arrivals	MOER, load	MOER	renewable wind penetration	outdoor temperature

some "demand" that is often affected by the behavior of unmodeled agents. For example, in `EVChargingEnv`, the demand is the amount of energy that needs to be delivered to EVs that have arrived at the charging network. This demand changed significantly at the start of the COVID-19 pandemic when EV drivers changed their driving behaviors (Figure 7.2).

2. *Shifts in environmental parameters.* Real-world environments are rarely static, and SustainGym environments reflect changing environment parameters due to temporal and/or climate changes. For example, a battery storage controller for `ElectricityMarketEnv` makes decisions to minimize marginal CO_2

emissions, but the distribution of CO_2 emissions varies over time as power plants are added to or removed from the electric grid (Figure 7.1).

Notably, the distribution shifts reflected in SustainGym are unlike the "sim-to-real" or offline-vs-online RL distribution shifts that have been more commonly studied in the literature. The sim-to-real distribution shift comes from imperfect modeling of the environment, whereas the offline-to-online RL distribution shift is caused by a change in the policy used to generate trajectories. In contrast, the exogenous distribution shifts in SustainGym are not due to imperfect environments nor policy mismatches, but rather more fundamental changes in the transition dynamics of the Markov decision processes. Note that only the transition dynamics experience distribution shift; the state space, action space, and reward functions do not change.

Two other similar lines of work to the distribution shifts in SustainGym are non-stationary RL environments [194] and distributionally robust RL [247]. However, whereas nonstationary RL typically evaluates an RL agent's performance over the course of a changing environment, SustainGym benchmarks RL agents' ability to generalize to new (unseen) distribution shifts. Distributionally robust RL generally assumes that the set of environmental distributions are known at training time, which is not necessarily the case for the settings considered in SustainGym.

In addition to modeling realistic distribution shifts, SustainGym is distinctive for its inclusion of multi-agent interactions, physical constraints, and mixtures of discrete and continuous actions, as summarized in Table 7.1.

To demonstrate the use of the SustainGym, we perform experiments with off-the-shelf RL algorithms. We find that these algorithms have mixed performance on SustainGym. Furthermore, we show that distribution shifts may reduce the performance of these algorithms significantly, demonstrating a need for more robust algorithms. Finally, comparisons against non-RL baselines and oracles show that RL has significant room for improvement.

The main text of this chapter summarizes key design choices and experimental observations for SustainGym. Details can be found in Section 7.6. Code, licenses, and instructions for using SustainGym can be found on GitHub.[1]

Related Work. Prior work related to SustainGym includes ConservationGym, which focuses on ecological applications [145], PowerGridWorld for power system modeling and simulation [34], and CityLearn for simulation of demand response and

[1] https://github.com/chrisyeh96/sustaingym/

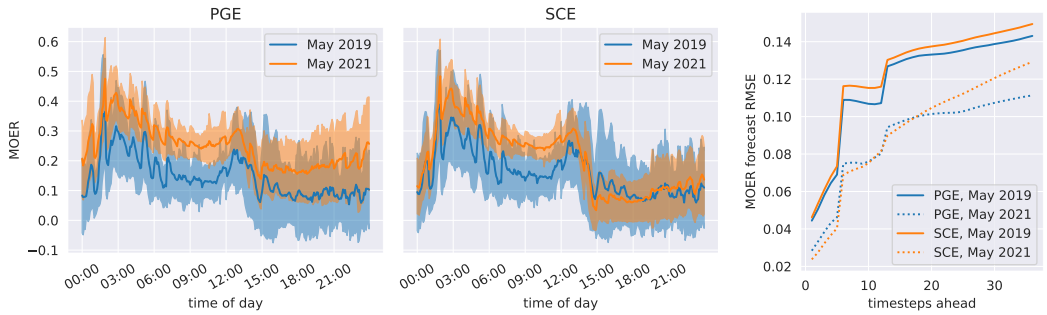


Figure 7.1: (left) MOER values from two different regions (a.k.a. "balancing authorities") in California, Pacific Gas & Electric (PGE) and Southern California Edison (SCE). Solid line is the mean MOER over all days in a month at a given time of day. Shaded region is ±1 std. dev. (right) MOER forecast error increases with the forecast horizon.

urban energy management [273], among others. RL environments and algorithms for both EV charging [149, 126, 1] and electricity markets [280, 298, 49] have also been released. Compared to these works, the unique aspects of SustainGym are its focus on tracking estimated CO_2 emissions and its ability to test RL algorithms in settings with challenging distribution shifts, physical constraints, and interactions between multiple agents. We expect SustainGym to serve as a benchmark for the progress of RL algorithm development for sustainable energy systems.

7.2 Environments

This section introduces the 5 environments in SustainGym and summarizes their design choices.

Marginal CO_2 emissions. Three environments (`EVChargingEnv`, `ElectricityMarketEnv`, `DatacenterEnv`) impose a cost P_{CO_2} (in \$/kgCO_2) on the simulated CO_2 emissions induced by the actions of an agent as a result of changes in electricity consumption. To do so, our environments use data on California's historical marginal operating emissions rate (MOER, in kgCO_2/kWh), which is the increase in CO_2 emissions per increase in energy demand. The MOER at time t is denoted $m_t \in \mathbb{R}_+$, and the forecasts generated at time t for the next k time steps are denoted $\hat{m}_{t:t+k-1|t} \in \mathbb{R}^k$. By default, we use $k = 36$. Figure 7.1 shows how MOER values and their forecasts vary across time and between different regions in California.

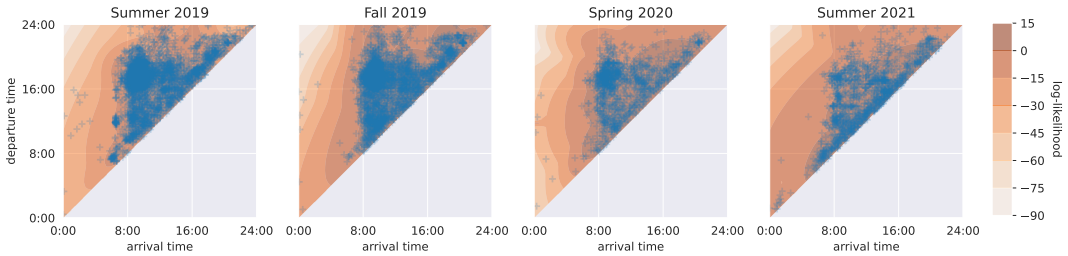


Figure 7.2: EV arrival vs. departure times for the Caltech EV charging network. Historical data is in blue, and log-likelihood contours from a 30-component GMM are in orange. The distribution of EV arrival and departure times changed noticeably when the COVID-19 pandemic began in early 2020.

`EVChargingEnv`

`EVChargingEnv` uses ACNSim [149] to simulate the charging of EVs based on actual data gathered from EV charging networks between fall 2018 and summer 2021 [148, 147]. ACNSim is a "digital twin" of actual EV charging networks at Caltech and JPL, which have $n = 54$ and 52 charging stations (abbrv. EVSEs, Electric Vehicle Supply Equipment), respectively. ACNSim accounts for nonlinear EV battery charging dynamics and unbalanced 3-phase AC power flows, and is thus very realistic. ACNSim (and therefore `EVChargingEnv`) can be extended to model other charging networks as well. When drivers charge their EVs, they provide an estimated time of departure and amount of energy requested. Because of network and power constraints, not all EVSEs can simultaneously provide their maximum charging rates (a.k.a. "pilot signals").

Each episode starts at midnight and runs at 5-minute time steps for 24 hours. At each time step, the agent simultaneously decides all n EVSE pilot signals to be executed for the duration of that time step. Its objective is to maximize charge delivery while minimizing carbon costs and obeying the network and power constraints.

Observation Space. An observation at time t is $s(t) = (t, d, e, m_{t-1}, \hat{m}_{t:t+k-1|t})$. $t \in [0, 1]$ is the fraction of day. $d \in \mathbb{Z}^n$ is estimated remaining duration of each EV (in # of time steps). $e \in \mathbb{R}^n_+$ is remaining energy demand of each EV (in kWh). If no EV is charging at EVSE i, then $d_i = 0$ and $e_i = 0$. If an EV charging at EVSE i has exceeded the user-specified estimated departure time, then d_i becomes negative, while e_i may still be nonzero.

Action Space. `EVChargingEnv` exposes a choice of discrete actions $a(t) \in \{0, 1, 2, 3, 4\}^n$, representing pilot signals scaled down by a factor of 8, or con-

tinuous actions $a(t) \in [0,1]^n$ representing the pilot signal normalized by the maximum signal allowed M (in amps) for each EVSE. Physical infrastructure in a charging network constrains the set $\mathcal{A}_t$ of feasible actions at each time step t [147]. Furthermore, the EVSEs only support discrete pilot signals, so $\mathcal{A}_t$ is nonconvex. To satisfy these physical constraints, `EVChargingEnv` can project an agent's action $a(t)$ into the convex hull of $\mathcal{A}_t$ and round it to the nearest allowed pilot signal, resulting in final normalized pilot signals $\tilde{a}(t)$. ACNSim processes $\tilde{a}(t)$ and returns the actual charging rate $M\bar{a} \in \mathbb{R}_+^n$ (in amps) delivered at each EVSE, as well as the remaining demand $e_i(t+1)$.

Reward Function. The reward function is a sum of three components: $r(t) = p(t) - c_V(t) - c_C(t)$. The profit term $p(t)$ aims to maximize energy delivered to the EVs. The constraint violation cost $c_V(t)$ penalizes network and power constraint violations. Finally, the CO_2 emissions cost $c_C(t)$, which is a function of the MOER m_t and charging action, aims to reduce emissions by encouraging the agent to charge EVs when the MOER is low.

Distribution Shift. `EVChargingEnv` supports real historical data as well as data sampled from a 30-component Gaussian Mixture Model (GMM) fit to historical data. We fitted GMMs to 4 disjoint historical periods, as defined in [158]. Figures 7.2 and 7.6 show the distribution of arrival and departure times in each of these 4 periods, for both the historical data as well as the GMM log-likelihoods. From these figures, it is evident that the pattern of user arrival and departure times changes over time, with the most drastic shift happening between Fall 2019 and Spring 2020, which is when the COVID-19 pandemic began.

Multiagent Setting. The multiagent setting features n agents, each deciding the pilot signal for a single EVSE. The reward is split evenly among the agents. Each agent obtains the global observation, except that the estimated remaining durations and energy demands for other EVSEs are delayed by t_d time steps.

`ElectricityMarketEnv`

`ElectricityMarketEnv` simulates a realtime electricity market for 33 generators and 1 80MWh battery storage system connected on a 24-bus congested transmission network based on the widely-used IEEE RTS-24 test case [255], with 5-minute settlements and load data from IEEE RTS-GMLC [20]. While `Electricity MarketEnv` is not modeled after any particular real-world transmission network, the

RTS-GMLC electricity load profile was designed to be representative of a modern transmission network located in the southwestern U.S.

All participants submit bids to the market operator (MO) at every time step. Based on the bids, the MO solves the multi-timestep security-constrained economic dispatch (SCED) problem which determines the price and amount of electricity purchased from (or sold by) each generator and battery to meet realtime electricity demand. Each episode runs for 1 day, with 5-minute time intervals. The agent controls the battery system and is rewarded for submitting bids that result in charging (buy) when prices are low, and discharging (sell) when prices and CO_2 emissions are high, thus performing price arbitrage.

Observation Space. An observation is

$$s(t) = (t, e, a(t-1), x_{t-1}, p_{t-1}, l_{t-1}, \hat{l}_{t:t+k-1}, m_{t-1}, \hat{m}_{t:t+k-1|t}).$$

$t \in [0, 1]$ represents the time of day. $e \in \mathbb{R}_+$ is the agent's battery level (in MWh). $a(t-1) \in \mathbb{R}_+^{2\times k}$ is the previous action taken. $x_{t-1} \in \mathbb{R}$ is the previous dispatch (in MWh) asked of the agent. $p_{t-1} \in \mathbb{R}_+$ is the previous price experienced by the agent (in \$/MWh). $l_{t-1} \in \mathbb{R}_+$ is the previous demand experienced by the agent (in MWh), while $\hat{l}_{t:t+k-1} \in \mathbb{R}^k$ is the forecasted demand for the next k steps.

Action Space. An agent action is a bid $a(t) = (a^c, a^d) \in \mathbb{R}_+^k \times \mathbb{R}_+^k$, representing prices (\$/MWh) that the agent is willing to pay (or receive) for charging (or discharging) per MWh of energy, for the next $k+1$ time steps starting from time t. The generators are assumed to always bid their fixed true cost of generation. The environment solves the optimal dispatch problem to determine the electricity price p_t and the agent's dispatch $x_t \in \mathbb{R}$, which is the amount of energy that the agent is obligated to sell into or buy from the grid within the next time step. The dispatch in turn determines the storage system's next energy level. We also provide a wrapper that discretizes the action space into 3 actions only: charge, do nothing, or discharge.

Reward Function. The reward function encourages the agent to maximize profit from charging decisions while minimizing associated carbon emissions. It is a sum of three components: $r(t) = r_R(t) + r_C(t) - c_T(t)$. The revenue term $r_R(t) = p_t x_t$ is the immediate revenue from the dispatch. The CO_2 emissions reward term $r_C(t) = P_{CO_2} m_t x_t$ represents the price of CO_2 emissions displaced or incurred by the battery dispatch. The terminal cost $c_T(t)$, which is nonzero only when $t = T$, encourages the battery to have the same energy level at the end of the day as when it

started. We also provide an option to delay all reward signals until the terminal time (intermediate rewards are 0).

Distribution Shift. Distribution shift for `ElectricityMarketEnv` comes from changes in both electricity demand and MOER profiles between summer and winter months.

`DatacenterEnv`

`DatacenterEnv` is a simulator for carbon-aware job scheduling in datacenters, which aims to reduce the carbon emissions associated with electricity usage in a datacenter. Carbon-aware job scheduling is premised upon two facts: (i) a significant fraction of a datacenter's workload (*e.g.*, up to 50% in some Google datacenters [274, 270]) is comprised of low priority jobs whose execution can be delayed, and (ii) the carbon intensity of the electric grid fluctuates predictably over time. Therefore, if the execution of low priority workload is delayed to a time of day with "greener" energy, the datacenter's carbon emissions can be minimized.

`DatacenterEnv` is loosely modeled after a Google datacenter. We assume that jobs are scheduled according to a priority queue, with jobs spread evenly across the available machines. Following Radovanovic et al. [214], we implement workload execution delay by artificially limiting the total datacenter capacity with a *virtual capacity curve* (VCC) at each time step. If more jobs are enqueued than the VCC permits, then the jobs must wait until the VCC is raised high enough to allow the jobs to run. Simulation is carried out by replaying a sample of real job traces from a Google cluster from May 2019 [270]. One timestep in the environment corresponds to one hour, and each episode lasts the whole month.

Observation Space. An observation $s(t) = (a(t-1), d_t, n, \hat{m}_{t:t+23|t}) \in \mathbb{R}^{27}$ contains the active VCC $a(t-1)$ set from the previous time step, currently running compute load d_t, number of jobs waiting to be scheduled n, as well as the forecasted MOER for the next 24h $\hat{m}_{t:t+23|t}$.

Action Space. At time t, the agent sets the VCC, $a(t) \in [0, 1]$, for the next time step. This action denotes the *fraction* of the datacenter's maximum capacity allowed to be allocated by the scheduler.

Reward Function. The reward consists of two components that encourage the agents to trade-off between scheduling more jobs and reducing associated carbon emissions. The first component penalizes the agent when jobs are scheduled more than 24h after they were originally submitted. The second component is a carbon

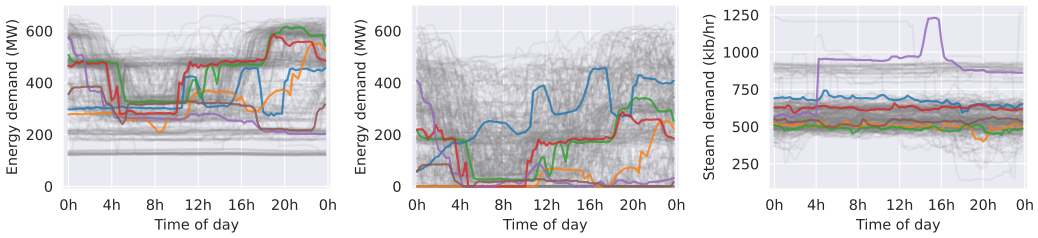


Figure 7.3: Electricity demand without (left) and with wind (middle), and steam demand (right) on a 15 minute basis for 253 days in `CogenEnv` dataset, with 6 random traces highlighted for each.

emissions cost. Formally, the reward is specified as

$$r(t) = d_t \cdot m_t + \mathbf{1}_{[t \bmod 24=0]} \max\left(0, 0.97 w_t - C \sum_{h=0}^{23} a(t-h)\right)$$

where $d_t \cdot m_t$ is the carbon emissions, C is the datacenter's maximum capacity, and w_t is the total job-hours of enqueued jobs on that day.

Distribution Shift. The distribution shift in `DatacenterEnv` comes from changes in the MOER between 2019 and 2021.

`CogenEnv`

`CogenEnv` simulates the operation of a combined cycle gas power plant tasked with meeting local steam and energy demand. Conventional dispatchable generators suffer decreased efficiency as a result of frequent ramping, posing a particular challenge as increasing penetrations of variable renewables necessitate larger and more frequent ramps to ensure supply-demand balance. Thus, optimal operation of cogeneration resources requires balancing the competing objectives of minimizing fuel use, anticipating future ramp needs, and ensuring delivery of sufficient energy and steam to the grid.

While `CogenEnv` models a specific combined cycle gas generation plant in the U.S. (anonymized and location withheld for security reasons), the basic environment setup is a representative prototype of more general dispatchable resource generation control tasks, due to its complexity (the number of variables, mixed continuous/binary decisions, complementary trains of the plant all needing to be controlled together). In addition, the environment is readily modifiable to accommodate other cost structures (*e.g.*, changing the relative magnitude of the constraint penalties vs. the ramping cost).

Observation Space. An observation takes the form

$$s(t) = (\tau, a(t-1), T_{t:t+k}, P_{t:t+k}, H_{t:t+k}, d^p_{t:t+k}, d^q_{t:t+k}, \pi^p_{t:t+k}, \pi^f_{t:t+k}),$$

where $\tau = t/96$ is the time (normalized by number of 15 minute intervals in a day), $a(t-1)$ is the agent's previous action, $T_{t:t+k}, P_{t:t+k}$, and $H_{t:t+k}$ are current and k forecast steps of temperature, pressure, and relative humidity, respectively, $d^p_{t:t+k}$ and $d^q_{t:t+k}$ are current and k forecast steps of electricity and steam demand, respectively, and $\pi^p_{t:t+k}$ and $\pi^f_{t:t+k}$ are current and k forecast steps of electricity and fuel price, respectively.

Action Space. The action space is a vector $a(t) \in \mathbb{R}^{15}$ specifying dispatch setpoints and other auxiliary variables for all turbines in the plant. Specifically, for each of three gas turbines, the agent specifies (a) a scalar turbine electricity output, (b) a scalar heat recovery steam flow, (c) a binary evaporative cooler switch setting, and (d) a binary power augmentation switch setting. In addition, for the steam turbine, the agent specifies (a) a scalar turbine electricity output, (b) a scalar steam flow through the plant condenser, and (c) an integer number of cooling tower bays employed.

Reward Function. The reward function is comprised of three components:

$$r(t) = -\left(r_f(a(t); T_t, P_t, H_t) + r_r(a(t); a(t-1)) + r_c(a(t); d^p_t, d^q_t)\right).$$

$r_f(a(t); T_t, P_t, H_t)$ is the generator fuel consumption in response to dispatch $a(t)$. $r_r(a(t); a(t-1))$ is the ramp cost, captured via an ℓ_1 norm penalty for any change in generator electricity dispatch between consecutive actions. $r_c(a(t); d^p_t, d^q_t)$ is a constraint violation penalty, penalizing any unmet electricity and steam demand, as well as any violation of the plant's dynamic operating constraints. The sum of these three components is negated to convert costs to rewards.

Distribution Shift. `CogenEnv` considers distribution shifts in the renewable generation profiles, and specifically, increasing penetration of wind energy. This increased variable renewable energy on the grid necessitates more frequent ramping in order to meet electricity demand, and may pose a challenge for RL algorithms trained on electricity demand traces without such variability.

Multiagent Setting. The multiagent setting treats each turbine unit (each of the three gas turbines and the steam turbine) as an individual agent whose action is the turbine's electricity dispatch decision and auxiliary variable settings. The negative reward of each agent is the sum of the corresponding turbine unit's fuel consumption,

ramp cost, and dynamic operating constraint penalty, as well as a shared penalty for unmet electricity and steam demand that is split evenly across agents. All agents observe the global observation.

`BuildingEnv`

`BuildingEnv` considers the control of the heat flow in a multi-zone building so as to maintain a desired temperature setpoint. Building temperature simulation uses first-principled physics models. Users can either choose from a pre-defined list of buildings (Office small, School primary, Apartment midrise, and Office large) and three climate types and cities (San Diego, Tucson, New York) provided by the Department of Energy (DoE) Building Energy Codes Program [207] or define a customized `BuildingEnv` environment by importing any self-defined EnergyPlus building models. Each episode runs for 1 day, with 5-minute time intervals ($H = 288$, $\tau = 5/60$ hours).

Observation Space. For a building with M indoor zones, the state $s(t) \in \mathbb{R}^{M+4}$ contains observable properties of the building environment at timestep t:

$$s(t) = (T_1(t), \ldots, T_M(t), T_{\mathrm{E}}(t), T_{\mathrm{G}}(t), Q^{\mathrm{GHI}}(t), \bar{Q}^{\mathrm{p}}(t)),$$

where $T_i(t)$ is zone i's temperature at time step t, $\bar{Q}^{\mathrm{p}}(t)$ is the heat acquisition from occupant's activities, $Q^{\mathrm{GHI}}(t)$ is the heat gain from the solar irradiance, and $T_{\mathrm{G}}(t)$ and $T_{\mathrm{E}}(t)$ denote the ground and outdoor environment temperature. In practice, the agent may have access to all or part of the state variables for decision-making depending on the sensor setup. Note that the outdoor/ground temperature, room occupancy, and heat gain from solar radiance are time-varying uncontrolled variables from the environment.

Action Space. The action $a(t) \in [-1, 1]^M$ sets the controlled heating supplied to each of the M zones, scaled to $[-1, 1]$.

Reward Function. The objective is to reduce energy consumption while keeping the temperature within a given comfort range. The default reward function is a weighted ℓ_2 reward, defined as

$$r(t) = -(1 - \beta) \, \|a(t)\|_2 - \beta \|T^{\mathrm{target}}(t) - T(t)\|_2$$

where $T^{\mathrm{target}}(t) = [T_1^{\mathrm{target}}(t), \ldots, T_M^{\mathrm{target}}(t)]^\top$ are the target temperatures and $T(t) = [T_1(t), \ldots, T_M(t)]^\top$ are the actual zonal temperatures. `BuildingEnv` also allows users to customize reward functions by changing the weight term β or the parameter

Table 7.2: Distribution shift experiments

Environment	What shifts	Original setting	Shifted setting
`EVChargingEnv`	EV sessions, MOER	Summer 2019	Summer 2021
`DatacenterEnv`	MOER	May 2019	May 2021
`CogenEnv`	Wind penetration	0 MW wind	300 MW wind
`BuildingEnv`	Ambient temperature	Summer 2004	Winter 2003

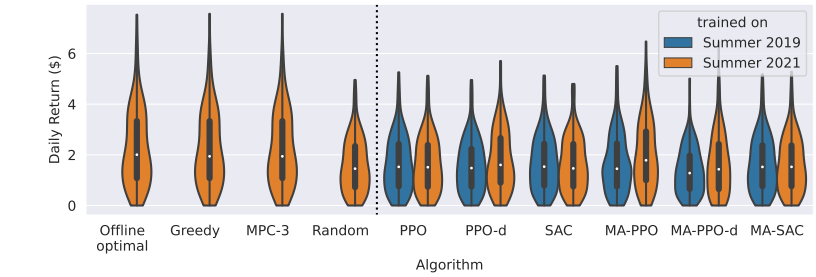
p defining the ℓ_p norm. Users can customize the reward function to consider CO_2 emissions and temperature constraints such as upper and lower temperature bounds.

Distribution Shift. `BuildingEnv` features distribution shifts in the ambient outdoor temperature profile T_E which varies with different seasons. `BuildingEnv` supports the distribution shifts due to the variation of seasons, located cities of the buildings, and can examine the challenges brought by such shifts in the RL environment.

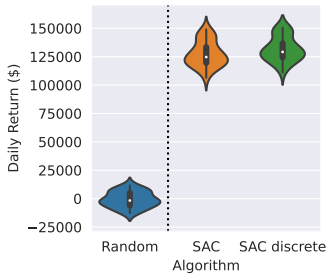
Multiagent Setting. In the multiagent setting for `BuildingEnv`, each agent controls the heating action for a single zone in the building. It must coordinate with other agents to maximize overall reward. Each agent obtains the same global observation and reward.

7.3 Experiments

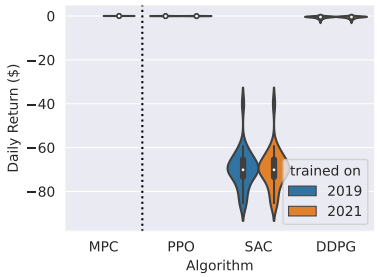
For each of the 5 environments in SustainGym, we implemented baseline non-RL algorithms as well as off-the-shelf RL algorithms trained using either RLLib [160] or Stable-Baselines3 (SB3) [215]. For most environments, we tested off-policy soft actor-critic (SAC) [102] and on-policy proximal policy optimization (PPO) [231]. Note that neither RLLib nor SB3 has an implementation of SAC that supports mixed discrete and continuous actions, as found in `CogenEnv`. For `EVChargingEnv`, we also tested multi-agent implementations of PPO and SAC, where the same policy is shared across agents. Non-RL algorithms tested include random policies and model predictive control (MPC), which is a model-based controller. Detailed descriptions of the implementations for each algorithm, including hyper-parameter tuning, are given in Section 7.6. Finally, to test distribution shift, we trained RL agents in both "original" and "shifted" environments, then compared their performance on the shifted environment, as described in Table 7.2.



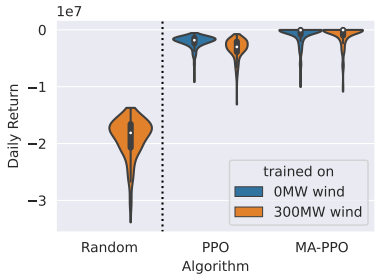
(a) `EVChargingEnv`



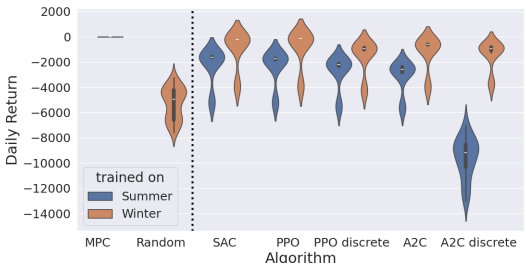
(b) `ElectricityMarketEnv`



(c) `DatacenterEnv`



(d) `CogenEnv`



(e) `BuildingEnv`

Figure 7.4: Experimental results for all 5 environments comparing performance on the shifted environment between RL algorithms trained on the original environment (blue) and RL algorithms trained on the shifted environment (orange). Policies using discretized actions are indicated with "-d", and multi-agent policies are prefixed with "MA-". For a complete description of the experiments, please see Section 7.6.

7.4 Discussion and Conclusion

Our experiments, shown in Figure 7.4, demonstrate a wide range of outcomes for off-the-shelf RL algorithms, with no single algorithm outperforming all the rest. In `EVChargingEnv`, for example, most of the RL algorithms perform no better than random actions, with the exception of multi-agent PPO with discrete actions. On `DatacenterEnv` and `BuildingEnv`, we notice a wider spread of returns across the different RL algorithms. In contrast, model-based MPC algorithms, where available, tend to perform more consistently than most RL algorithms.

In terms of distribution shift, we see a wide range of outcomes between agents trained on the original environments versus the shifted environments. Surprisingly, in `CogenEnv`, both single-agent and multi-agent policies trained on the shifted environment perform worse on the shifted environment than agents trained on the original environment. We believe this result may be due to the increased variability of shifted environment, making the shifted environment harder to learn in. In `DatacenterEnv`, the shift in MOER values shows essentially no effect on agent performance. In `EVChargingEnv`, agents trained on the shifted environment generally perform slightly better than agents trained on the original environment. In `BuildingEnv`, agents trained on the shifted environment perform much better.

These results highlight that the distribution shifts present in SustainGym environments provide substantial opportunities for future research, including robust RL algorithms [199] as well as online learning under distribution shift. Developing RL algorithms that are robust to these natural distribution shifts will be critical for deploying RL in the real-world high-impact sustainability settings such as those modeled by SustainGym environments.

Multi-agent RL. SustainGym is currently designed to support multi-agent RL in 3 environments, with the goal of upgrading all environments with multi-agent support in the future. In the two environments for which we tested multi-agent RL policies (`EVChargingEnv` and `CogenEnv`), the multi-agent PPO (MA-PPO) policies out-performed all other RL policies. We suspect that this may be because the action spaces in these environments factorize well across agents, so the multi-agent policies can learn more efficiently. Furthermore, we suspect that multi-agent policies may have the potential of performing better under distribution shift, since their environments are naturally non-stationary during training. We notice this to be true for both `EVChargingEnv` and `CogenEnv`: of the RL policies trained on the original

environments, the multi-agent policies performed best when tested on the shifted environment.

Future work. SustainGym is under active development, with several key directions of future work:

- Comprehensive support for multi-agent RL. Currently, only 3 out of the 5 environments support multi-agent RL. We are working to extend the two other environments `ElectricityMarketEnv` and `DatacenterEnv` to the multi-agent RL setting. For `ElectricityMarketEnv`, we plan on introducing multiple batteries into the same transmission network, each controlled by separate competing agents. This will be the first competitive multi-agent RL environment in SustainGym. (All other environments feature cooperative multi-agent RL.) For `DatacenterEnv`, we plan on introducing multiple datacenters spread across geographic regions to enable both temporal and geographic carbon-aware load shifting. Each datacenter would be its own agent.
- Different degrees of distribution shift. Currently, SustainGym environments feature a binary choice of distribution shift: an original environment, and a shifted environment. We plan on introducing more settings with varying degrees of distribution shift.
- More environments. We welcome new environment ideas and contributions to SustainGym and are working with potential collaborators to extend the scope of environments.

Limitations We conclude by acknowledging general limitations of SustainGym. First, SustainGym only captures very limited dimensions of sustainability (*i.e.*, energy and CO_2 emissions) and does not account for other aspects such as water usage and other pollutants associated with energy production. We welcome collaboration with experts in these other sustainability domains to help us improve the sustainability mission of SustainGym. Second, SustainGym is limited in the types of distribution shifts that are considered. Finally, while SustainGym environments have been designed to be reasonably representative of various sustainable energy settings, there is inevitably a gap between SustainGym simulations and actual hardware systems. A detailed discussion of the representativeness, generalizability, and limitations of each environment can be found in Section 7.6.

7.5 Metadata

Hosting and maintenance

SustainGym is hosted as a Python package on GitHub and PyPI. SustainGym is semantically versioned, and the version accompanying this publication is designated v1.0.

A guide for contributors is available in the GitHub repo.

The following authors are the maintainers for each environment in SustainGym:

- `EVChargingEnv`: Christopher Yeh
- `ElectricityMarketEnv`: Christopher Yeh
- `DatacenterEnv`: Christopher Yeh
- `CogenEnv`: Nicolas Christianson
- `BuildingEnv`: Chi Zhang

Licenses and responsibility

SustainGym as a whole is released under a CC BY 4.0 license.[2] However, the `Cogen Env` and accompanying code is released under a more restrictive CC BY-NC-SA 4.0 license,[3] per the terms of the model provider Enexsa. These licenses are available in our GitHub repo.

Intended uses

SustainGym is intended as a testbed for RL algorithms in sustainability-focused energy systems. While significant efforts have been undertaken to make the SustainGym environments closely match real-world systems, performance on SustainGym environments is not a guarantee of performance on real-world systems.

7.6 Environment and Experiment Details

EVChargingEnv

Assumptions. In addition to the description given in Section 7.2, `EVChargingEnv` makes the following assumptions:

- EVs staying overnight can be ignored. Upon the start of each episode, we assume the garage is empty. Analysis of historical traces on the adaptive charging network show that at most 12 cars at one time stay through midnight. Because this number is small compared to the number of stations, we do not

[2] https://creativecommons.org/licenses/by/4.0/
[3] https://creativecommons.org/licenses/by-nc-sa/4.0/

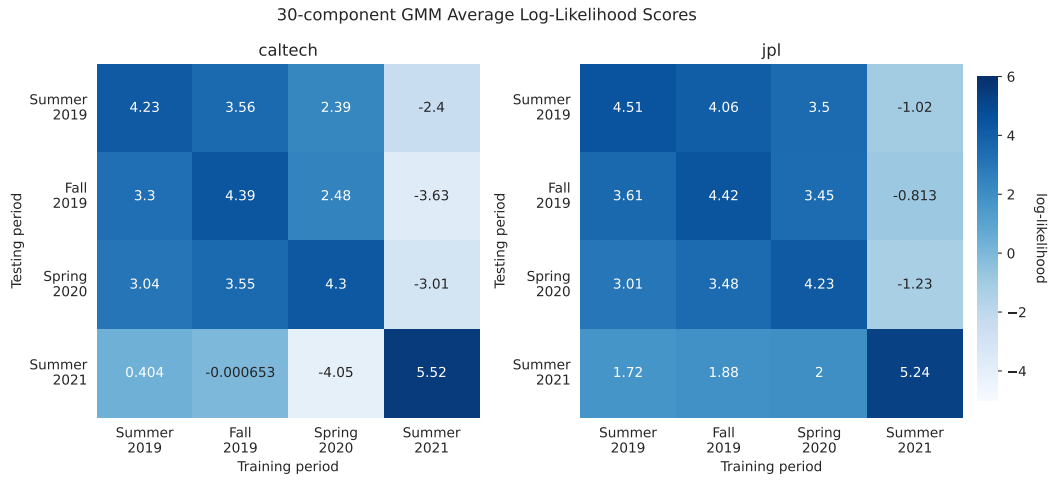


Figure 7.5: Log-likelihoods of data from testing periods (y-axis) for GMMs fitted on a training period (x-axis) using 30 components for the Caltech (left) and JPL (right) sites. A higher score implies a better fit. In all cases, GMMs scored highest on the period it was trained on. The significant drop in log-likelihood scores off-diagonal is indicative of distribution shift.

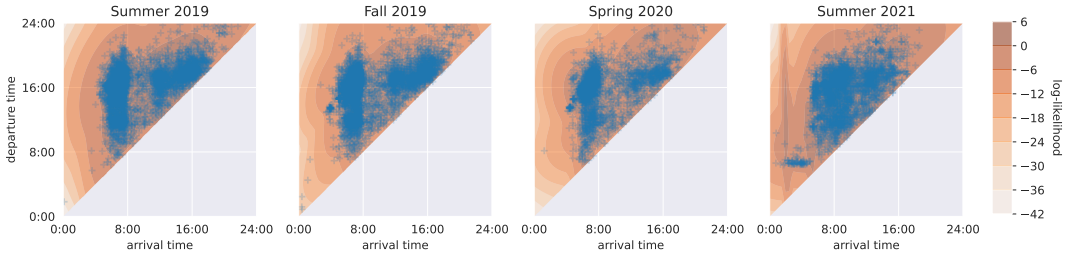


Figure 7.6: Like Figure 7.2, but for the JPL charging network.

expect the assumption of an empty garage at the start of the day to significantly impact the accuracy of the environment in representing real-world settings.

- All EVs have identical batteries. Because ACN-Data does not include data on the model of each EV, yet ACN-Sim models battery charging dynamics which vary based on battery capacity, we assume that all EVs contain 100 kWh batteries and come with an initial state of charge such that if the EV were charged to the amount of the user-requested energy, the battery would be full. Energies requested over 100 kWh are capped at 100 kWh. We use the `Linear2StageBattery` battery model within ACN-Sim.

Thus, each historical charging session includes time of EV arrival, estimated departure, actual departure, energy delivered, and EVSE ID.

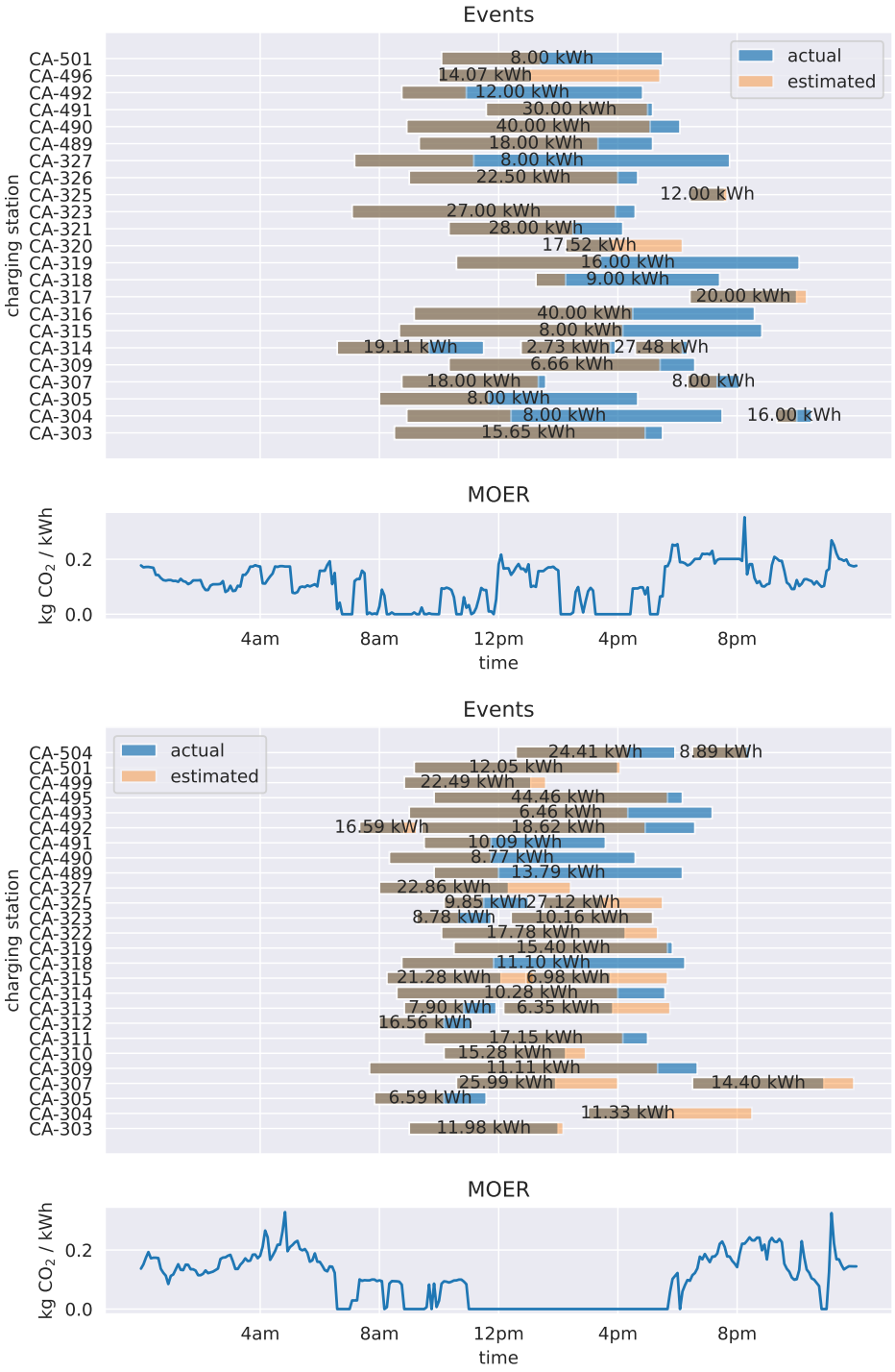


Figure 7.7: (top) A real full day's charging sessions from the Caltech charging network. (bottom) A randomly sampled trace of charging sessions from the GMMs trained on the Caltech historical data.

Feasible action space. Let $\mathcal{L}$ denote physical infrastructure resources (e.g., transformers or breakers) in the charging network. These infrastructure resources are characterized by (A, ϕ, c, v) where $A \in \mathbb{R}^{|\mathcal{L}| \times n}$ accounts for the charging network layout, $\phi \in \mathbb{R}^n$ is the voltage phase angle of each EVSE, $c \in \mathbb{R}^{|\mathcal{L}|}$ is the capacity limit for each resource, and $v \in \mathbb{R}_+$ is the EVSE voltage (in kV). Along with the demand at each EVSE, (A, ϕ, c) defines the time-dependent set of valid actions $\mathcal{A}_t$. Lastly, let $M = 32$ denote the maximum allowed pilot signal (in amps) for each EVSE.

When an agent gives the environment its desired normalized pilot signals $a(t) \in [0, 1]$, the projection into the convex hull of $\mathcal{A}_t$ is performed by solving the following convex optimization problem:

$$\begin{aligned}
\min_{\hat{a} \in \mathbb{R}^n} \quad & \|a(t) - \hat{a}\|_2 && && \text{(7.1a)}\\
\text{s.t.} \quad & 0 \le \hat{a}_i \le 1 && \forall i \in \{1, \dots, n\} && \text{(7.1b)}\\
& M \hat{a}_i v \tau \le e_i && \forall i \in \{1, \dots, n\} && \text{(7.1c)}\\
& \underbrace{\left| \sum_{i=1}^{n} A_{li} M \hat{a}_i e^{j\phi_i} \right|}_{|I_l|} \le c_l && \forall l \in \mathcal{L} && \text{(7.1d)}
\end{aligned}$$

Here, $j = \sqrt{-1}$ is the imaginary number and I_l is the aggregate current through constraint $l \in \mathcal{L}$. The quantity $M \hat{a}_i v \tau$ computes the energy (in kWh) to be charged from EVSE i during the next time step. The continuous pilot signals $M\hat{a}$ are then rounded to the set of discrete pilot signals supported by each EVSE, resulting in the final pilot signals $M\tilde{a}(t)$.

Reward function. The profit term $p(t)$ aims to maximize the amount of energy delivered to the EVs. Let $\pi \in \mathbb{R}_+$ denote a fixed marginal profit (in \$/kWh) that the EV charging network earns for each unit of energy delivered and $v \in \mathbb{R}_+$ be the voltage (in kV) of the charging network. Then,

$$p(t) = \pi \sum_{i=1}^{n} M \bar{a}_i v \tau = \pi M \bar{a}^\top \mathbf{1} v \tau$$

where $\mathbf{1}$ denotes a vector of all ones. The constraint violation cost $c_V(t)$ aims to reduce physical constraint violations and encourage the agent's action $a(t)$ to be in

$\mathcal{A}_t$. We penalize violation costs with a weight λ_V (in \$/kWh), so

$$c_V(t) = \lambda_V \sum_{l \in \mathcal{L}} \max\{I_l(t) - c_l, 0\} v\tau$$

where I_l is the electrical current and c_l is the maximum current capacity at resource l. By default, we set $\lambda_V = 0.01$.

Finally, the CO_2 emissions cost $c_C(t)$ aims to reduce emissions by encouraging the agent to charge EVs when the MOER is low. We have

$$c_C(t) = P_{CO_2} m_t \sum_{i=1}^{n} M\bar{a}_i v\tau = P_{CO_2} m_t M\bar{a}^\top \mathbf{1} v\tau$$

Model predictive control (MPC). As a baseline non-RL algorithm, we consider a model predictive control (MPC) controller similar to what is proposed in [147]. Let $w \leq k$ denote the length of the lookahead window (up to the number of MOER forecast steps k). Then, at every time step t, the MPC controller solves the following optimization problem:

$$\begin{aligned}
\max_{a^0,\ldots,a^{w-1} \in \mathbb{R}^n} \quad & \sum_{k=0}^{w-1} (\pi - P_{CO_2} \hat{m}_{t+k|t}) M a^{k\top} \mathbf{1} v\tau && && \text{(7.2a)} \\
\text{s.t.} \quad & 0 \preceq a^k \preceq 1 && \forall k \in \{0, \ldots, w-1\} && \text{(7.2b)} \\
& \sum_{k=0}^{w-1} M a^k v\tau \preceq e && && \text{(7.2c)} \\
& a_i^k = 0 && \forall i \in \{1, \ldots, n\},\ k \geq d_i && \text{(7.2d)} \\
& \underbrace{\left| \sum_{i=1}^{n} A_{li} M a_i^k e^{j\phi_i} \right|}_{|I_l|} \leq c_l, && \forall l \in \mathcal{L},\ k \in \{0, \ldots, w-1\} && \\
& && && \text{(7.2e)}
\end{aligned}$$

The optimization problem plans pilot signals for the next w time steps to maximize revenue and minimize forecasted carbon emissions cost. (7.2c) ensures that the pilot signals do not over-charge the EVs. (7.2d) prevents charging EVs after their estimated departure times. (7.2e) is the usual infrastructure constraint.

Only the first planned pilot signal Ma^0 is used. It is rounded to the set of discrete pilot signals supported by each EVSE, resulting in the final pilot signal $M\tilde{a}(t)$. On

the next time step, the MPC algorithm resolves the optimization problem. Figure 7.4a shows the distribution of returns for "MPC-3", which sets $w = 3$.

Training. The entire summer 2021 period [158] was selected as the testing period. We performed three splits: 1) by RL algorithm (PPO and multi-agent PPO vs. SAC and multi-agent SAC), 2) continuous vs. discrete action space, and 3) training data generated by GMMs based on out-of-distribution data (summer 2019) vs. in-distribution data (summer 2021). In each of the four cases, we used action projection during training and testing, eliminating costs of network violations. We tested 3 learning rates for each algorithm: PPO (5e-6, 5e-5, 5e-4) and SAC (1e-2, 1e-3, 1e-4). These learning rates were chosen as the default RLLib learning rates for each algorithm, scaled by factors of 10, 1, and 0.1. For each learning rate, three different random seeds were tested, and the model with the best training performance was selected.

Representativeness and Generalizability. `EVChargingEnv`, as mentioned in Section 7.2, is based on the actual EV charging networks in place at Caltech and the Jet Propulsion Laboratory (JPL), both located in Pasadena, California, U.S.A. `EVChargingEnv` uses a "digital twin" of these networks, called ACN-Sim [149], as well as real historical EV charging data [148], in the simulation of the RL environment. While different EV charging networks may have different constraints, the adaptive EV charging problem is similar across all networks (*i.e.*, deciding pilot signals for EVSEs to maximize energy delivery while minimizing costs and satisfying network and power constraints). In this way, `EVChargingEnv` is very representative of this type of control task. Furthermore, the code for ACN-Sim is open-source and well-documented,[4] which allows `EVChargingEnv` to be extended to model other charging networks as well.

Limitations. As mentioned under "Assumptions" above, `EVChargingEnv` uses some simplifying assumptions, in part because of limited data availability. Furthermore, the reward function in `EVChargingEnv` currently assigns a fixed marginal profit per unit of energy delivered to an EV, whereas larger EV charging networks are often affected by time-varying wholesale electricity prices. Finally, electricity grid failures and communication failures are not modeled in `EVChargingEnv`.

[4] https://acnportal.readthedocs.io/

ElectricityMarketEnv

System Model and Motivation. `ElectricityMarketEnv` aims to simulate grid-scale battery storage systems participating in an energy market for a regional transmission network. A battery has two objectives: make profit from price arbitrage (buy low, sell high) while minimizing its associated CO_2 emissions. The CO_2 emissions are accounted for when both charging and discharging:

- When a battery charges energy from the grid, it incurs the CO_2 emissions associated with grid's current marginal emissions rate (MOER).
- When a battery discharges energy into the grid, it offsets other generators that would have had to supply more electricity, thereby reducing CO_2 emissions commensurate with the grid's current MOER.

Network congestion refers to physical limits on transmission line capacity that prevent a generator from supplying electricity to a load at an arbitrary bus (a.k.a. "node") on the network. Such constraints lead to inefficiencies but are present in almost all real-world transmission networks. Thus, modeling these transmission constraints is important for ensuring electricity grid reliability.

The parameters of `ElectricityMarketEnv` are listed in Table 7.3. The default environment is based on the Region 1 in the IEEE Reliability Test System (RTS) [20], with 5-minute settlements ($\delta = 5/60$ hours), an episode length of $T = 288$ (1 day), and a lookahead horizon of $h = 36$ steps (3 hours). The IEEE RTS network features $N = 24$ nodes (a.k.a. buses) connected with $M = 38$ lines. Connected to the network are $N_G = 33$ conventional thermal generators and $N_D = 17$ loads. The network is slightly modified to add a single ($N_B = 1$) 80MWh battery system located at bus 11, with maximum charge/discharge rate $\overline{\mathbf{b}^{\mathrm{c}}} = \overline{\mathbf{b}^{\mathrm{d}}} = 20$MW, and initial and final state of charge (SOC) $\underline{\mathbf{x}} = 0$, $\mathbf{x}^0 = \mathbf{x}^f = 40$MWh. Charge/discharge efficiency is set to $\eta^{\mathrm{c}} = \eta^{\mathrm{d}} = 0.95$, and the minimum generation limit for all generators is changed to $\underline{g} = 0$.

We assume that an independent system operator (ISO) solves standard security-constrained economic dispatch (SCED) with the DC (decoupled) power flow equations. "Security-constrained" means that line limits are respected. The formulation has the following features:

- 3 classes of market participants: generators, loads, and storage
- DC PF equations
- Line flows and constraints

- Linear cost functions
- Nodal demand timeseries

The formulation lacks the following features that may be considered important in a practical implementation of SCED:

- Unit commitment for thermal units
- Line and generator outage contingencies, *e.g.*, $N-1$ security
- Operating reserves
- Piecewise linear and quadratic generator cost functions
- Flexible, curtailable load
- Renewable generators
- Generator ramping constraints

Battery storage. There are N_B batteries in the system, of which only one is seeking to learn an optimal bidding strategy. Assign index $k = 1$ to this unit.

The bid must satisfy power bounds and non-simultaneity. The strategic battery must manage its own state of charge. The other $B-1$ batteries are assumed to be managed by the ISO. We do not consider the possibility of simultaneous charge/discharge.

The state of charge evolution of each battery is linear in charge/discharge efficiency parameters $\eta_i^{\mathrm{c}}, \eta_i^{\mathrm{d}}$. The charge/discharge limits and the SOC constraints are assumed to be known by the system operator for all units and are not part of the strategic agent's bid. The initial state of charge in each interval is given by the solution from the previous interval's lookahead optimization.

Generation and load. There are N_G generators in the system, with potentially multiple generators at each node. We do not consider ramping limits on generators or non-convexities from start-up and shutdown costs and constraints. The generators are constrained between their minimum ($\underline{\mathbf{g}} = 0$) and maximum production limits $\overline{\mathbf{g}}$. It is assumed that a feasible unit commitment exists for the given $\underline{\mathbf{g}}$ and $\overline{\mathbf{g}}$.

The load vector $\mathbf{d}_t \in \mathbb{R}^{N_D}$ is taken to be fixed (inflexible load). The demand prediction for $\tau > t$ is given by $\hat{\mathbf{d}}_\tau$, provided by an hourly day-ahead forecast.

Powerflow equations and line limits. The line susceptance matrix is $B = \mathrm{diag}(B_1, \ldots, B_M) \in \mathbb{R}_+^{M \times M}$ where $B_j \in \mathbb{R}$ is the susceptance of line j.

The matrix $C \in \{-1, 0, 1\}^{N \times M}$ is node-edge incidence matrix of the graph.

The slack bus of the network is assigned WLOG to node index $i = 1$. By convention the voltage angle of this node is fixed:

$$\theta_{1,t} = 0 \quad \forall t.$$

Given a vector of nodal real power injections $\mathbf{p}_t \in \mathbb{R}^N$, the power flows along lines in the network $\mathbf{f}_t \in \mathbb{R}^M$ is given by $\mathbf{f}_t = H\mathbf{p}_t$ with the generation shift matrix $H \in \mathbb{R}^{M \times N}$ defined as $H := BC^\top (CBC^\top)^\dagger$.

Economic dispatch problem. We implement a version of SCED called multi-interval lookahead real-time economic dispatch. In this version of the market clearing, the systems operator seeks to clear the market sequentially for each timestep (*e.g.*, every 5 mins), optimizing over the current interval t plus a lookahead horizon of h additional intervals. Only the solution from interval t is retained; the remaining h decisions are advisory. In systems with intertemporal constraints (*e.g.*, energy storage, unit commitment, ramping), it is necessary to perform this multi-interval dispatch to improve *ex-post* optimality and as well as to retain feasibility. In practice, North American ISOs all solve a version of multi-interval dispatch.

For interval t, the multi-interval SCED problem is

$$\min_{\mathbf{g}_t, \mathbf{b}_t^{\mathrm{c}}, \mathbf{b}_t^{\mathrm{d}}} \quad \delta \sum_{\tau=t}^{t+h} \left[\sum_{k=1}^{N_g} c_{k,\tau}^{\mathrm{g}} g_{k,\tau} + \sum_{k=1}^{N_b} c_{k,\tau}^{\mathrm{b,d}} b_{k,\tau}^{\mathrm{d}} - c_{k,\tau}^{\mathrm{b,c}} b_{k,\tau}^{\mathrm{c}} \right] \tag{7.3a}$$

$$\text{s.t.} \quad \Sigma[\mathbf{d}_\tau^\top, \mathbf{g}_\tau^\top, \mathbf{b}_\tau^{\mathrm{c}\top}, \mathbf{b}_\tau^{\mathrm{d}\top}]^\top = \mathbf{p}_\tau \qquad \forall \tau = t, \ldots, t+h \tag{7.3b}$$

$$\lambda_\tau \ \perp \ \mathbf{1}^\top \mathbf{p}_\tau \delta = 0 \qquad \forall \tau = t, \ldots, t+h \tag{7.3c}$$

$$\boldsymbol{\mu}_\tau^+, \boldsymbol{\mu}_\tau^- \ \perp \ |H\mathbf{p}_t| \leq \mathbf{f}^{\max} \qquad \forall \tau = t, \ldots, t+h \tag{7.3d}$$

$$\mathbf{d}_\tau = \hat{\mathbf{d}}_\tau \qquad \forall \tau = t, \ldots, t+h \tag{7.3e}$$

$$\underline{\mathbf{g}} \leq \mathbf{g}_\tau \leq \overline{\mathbf{g}} \qquad \forall \tau = t, \ldots, t+h \tag{7.3f}$$

$$\mathbf{0} \leq \mathbf{b}_\tau^{\mathrm{c}} \leq \overline{\mathbf{b}^{\mathrm{c}}} \qquad \forall \tau = t, \ldots, t+h \tag{7.3g}$$

$$\mathbf{0} \leq \mathbf{b}_\tau^{\mathrm{d}} \leq \overline{\mathbf{b}^{\mathrm{d}}} \qquad \forall \tau = t, \ldots, t+h \tag{7.3h}$$

$$\mathbf{x}_\tau = \mathbf{x}_{\tau-1} + \mathrm{diag}(\boldsymbol{\eta}^{\mathrm{c}}) \mathbf{b}_\tau^{\mathrm{c}} \delta - \mathrm{diag}(\boldsymbol{\eta}^{\mathrm{d}})^{-1} \mathbf{b}_\tau^{\mathrm{d}} \delta \qquad \forall \tau = t, \ldots, t+h \tag{7.3i}$$

$$\underline{\mathbf{x}} \leq \mathbf{x}_\tau \leq \overline{\mathbf{x}} \qquad \forall \tau = t, \ldots, t+h \tag{7.3j}$$

$$\mathbf{x}_{t-1} = \mathbf{x}_{t-1}^* \tag{7.3k}$$

$$\mathbf{x}_{t+h} = \mathbf{x}^{\mathrm{f}} \tag{7.3l}$$

Here, $\mathbf{x}_{t-1}^*$ is the state of charge decision from the previous interval. The constraint says that the current state of charge is whatever the battery was charged/discharged to in the previous dispatch.

In order to ensure that the battery is never simultaneously charging and discharging (*i.e.*, for every $k \in \{1, \ldots, N_b\}$ and $\tau \in \{t, \ldots, t+h\}$, at most one of $b_{k,\tau}^{\mathrm{c}}$ and $b_{k,\tau}^{\mathrm{d}}$ should be nonzero), we can instead formulate the problem as a mixed-integer linear program:

$$\min_{\mathbf{g}_t, \mathbf{b}_t^{\mathrm{c}}, \mathbf{b}_t^{\mathrm{d}}} \quad \delta \sum_{\tau=t}^{t+h} \left[\sum_{k=1}^{N_g} c_{k,\tau}^{\mathrm{g}} g_{k,\tau} + \sum_{k=1}^{N_b} c_{k,\tau}^{\mathrm{b,d}} b_{k,\tau}^{\mathrm{d}} - c_{k,\tau}^{\mathrm{b,c}} b_{k,\tau}^{\mathrm{c}} \right] \tag{7.4a}$$

$$\text{s.t.} \quad \Sigma[\mathbf{d}_\tau^\top, \mathbf{g}_\tau^\top, \mathbf{b}_\tau^{\mathrm{c}\top}, \mathbf{b}_\tau^{\mathrm{d}\top}]^\top = \mathbf{p}_\tau \qquad \forall \tau = t, \ldots, t+h \tag{7.4b}$$

$$\mathbf{1}^\top \mathbf{p}_\tau \delta = 0 \qquad \forall \tau = t, \ldots, t+h \tag{7.4c}$$

$$|H\mathbf{p}_t| \leq \mathbf{f}^{\max} \qquad \forall \tau = t, \ldots, t+h \tag{7.4d}$$

$$\mathbf{d}_\tau = \hat{\mathbf{d}}_\tau \qquad \forall \tau = t, \ldots, t+h \tag{7.4e}$$

$$\underline{\mathbf{g}} \leq \mathbf{g}_\tau \leq \overline{\mathbf{g}} \qquad \forall \tau = t, \ldots, t+h \tag{7.4f}$$

$$\mathbf{z}_\tau \in \{0,1\}^{N_b} \qquad \forall \tau = t, \ldots, t+h \tag{7.4g}$$

$$\mathbf{0} \leq \mathbf{b}_\tau^{\mathrm{c}} \leq \overline{\mathbf{b}^{\mathrm{c}}} \odot \mathbf{z}_\tau \qquad \forall \tau = t, \ldots, t+h \tag{7.4h}$$

$$\mathbf{0} \leq \mathbf{b}_\tau^{\mathrm{d}} \leq \overline{\mathbf{b}^{\mathrm{d}}} \odot (\mathbf{1} - \mathbf{z}_\tau) \qquad \forall \tau = t, \ldots, t+h \tag{7.4i}$$

$$\mathbf{x}_\tau = \mathbf{x}_{\tau-1} + \mathrm{diag}(\boldsymbol{\eta}^{\mathrm{c}})\mathbf{b}_\tau^{\mathrm{c}} - \mathrm{diag}(\boldsymbol{\eta}^{\mathrm{d}})^{-1}\mathbf{b}_\tau^{\mathrm{d}} \qquad \forall \tau = t, \ldots, t+h \tag{7.4j}$$

$$\underline{\mathbf{x}} \leq \mathbf{x}_\tau \leq \overline{\mathbf{x}} \qquad \forall \tau = t, \ldots, t+h \tag{7.4k}$$

$$\mathbf{x}_{t-1} = \mathbf{x}_{t-1}^* \tag{7.4l}$$

$$\mathbf{x}_{t+h} = \mathbf{x}^{\mathrm{f}} \tag{7.4m}$$

However, the mixed-integer linear program does not produce dual-variables the same way that the original linear program does. In order to recover nodal prices, we first solve the mixed-integer linear program to determine the optimal values for $\mathbf{z}_\tau$. Then, in the linear program, we replace $\overline{\mathbf{b}^{\mathrm{c}}}$ with $\overline{\mathbf{b}^{\mathrm{c}}} \odot \mathbf{z}_\tau$ and $\overline{\mathbf{b}^{\mathrm{d}}}$ with $\overline{\mathbf{b}^{\mathrm{d}}} \odot (\mathbf{1} - \mathbf{z}_\tau)$, and we solve the linear program to get the nodal prices from the dual variables. Although this procedure requires solving two optimization problems (first the MILP, and later the linear program), in practice solving the linear program is very fast, since the optimization variables can be initialized to their optimal values from the MILP.

Market clearing and settlement. The market clears when optimization problem (7.3) has a feasible solution, denoted $(\mathbf{g}_t^*, \mathbf{b}_t^{\mathrm{c}*}, \mathbf{b}_t^{\mathrm{d}*})$. The nodal vector of market clearing prices $\boldsymbol{\pi} \in \mathbb{R}^N$ (in \$/MWh) is defined by a function of dual variables $\lambda_t^*, \boldsymbol{\mu}_t^{+*}, \boldsymbol{\mu}_t^{-*}$:

$$\boldsymbol{\pi}_t := \lambda_t^* \mathbf{1} + H^\top (\boldsymbol{\mu}_t^{+*} - \boldsymbol{\mu}_t^{-*})$$

For each interval t, the settlement rule for generator k at node i is:

1. Generator k produces power $g_{k,t}^*$

2. Generator k receives revenue $\pi_{i,t} g^*_{k,t}$

The settlement rules for loads and batteries are analogous. When a battery is charging ($b^{\mathrm{c}}_{k,t} > 0$), it pays the nodal price; when it is discharging ($b^{\mathrm{d}}_{k,t} > 0$), it receives the nodal price.

Representativeness and Generalizability While `ElectricityMarketEnv` is not modeled after any particular real-world transmission network, it simulates the transmission network from the widely-used IEEE Reliability Test System (IEEE RTS-24) [255]. As the IEEE RTS-24 test case did not include electricity load data, we chose to incorporate load data from the recent 2019 IEEE RTS-GMLC update [20], which was designed to be representative of a modern transmission network located in the southwestern U.S., featuring a variety of renewable and distributed generators as well as representative electricity load profiles. According to industry experts, both the IEEE RTS-24 and IEEE RTS-GMLC are simplified but standard test cases. Furthermore, `ElectricityMarketEnv` has a modular design and is readily modified to simulate a particular network.

The multi-time-step security-constrained economic dispatch problem (SCED) implemented in `ElectricityMarketEnv` (7.3) is also representative of how market operators schedule generators and determine nodal prices in most electricity markets in the U.S.A.

Limitations Designing `ElectricityMarketEnv` to be easy to use necessitated some limitations in what could be modeled. First, the default IEEE RTS-24 network only features 24 buses, which is smaller than most real-world transmission networks. However, because each step of the environment requires solving a mixed-integer linear program, a larger test case would have significantly increased the amount of time taken in the environment and thus slowed down any RL training. Users who wish to test their RL algorithms in larger transmission networks are welcome to modify `ElectricityMarketEnv` for their needs.

Second, `ElectricityMarketEnv` is only based on data representative of modern electricity markets in the U.S.A., whereas many regions around the world still feature vertically-integrated energy monopolies and/or "traditional" electricity markets that may not necessarily solve a similar SCED optimization problem.

Finally, `ElectricityMarketEnv` only features distribution shifts in the form of changes in the marginal carbon emissions and load profiles between summer and

winter months. In the real-world, distribution shifts also occur when more generators are added to the network, when older generators are retired, and when the transmission network changes (*e.g.*, when new transmission lines are added or upgraded, or when transmission lines are taken offline by extreme weather events).

DatacenterEnv

System Model and Motivation. `DatacenterEnv` is inspired by the carbon-aware job scheduling approach adopted by Google's datacenters, as described in [214]. The Google approach can be summarized as follows. Each day, the system plans the 24-hour virtual capacity curve (VCC) for the next day by solving a constrained optimization problem whose objective is to minimize a weighted sum of expected carbon emissions and expected peak power consumption. The main constraint is that the VCC for the next day must sum to the 97th-percentile of the predicted total daily capacity requirement. Capacity is measured in terms of normalized CPU compute units. The VCC artificially limits the total datacenter capacity at each hour, forcing enqueued jobs to be scheduled later than they would otherwise run. The goal is to reduce the number of running jobs when the carbon intensity of the electrical grid (CO_2 emissions per electrical energy used) is high, and run more jobs when the carbon intensity is low. By time-shifting jobs, the datacenter is able to reduce its CO_2 emissions.

The VCC-based approach to carbon-aware job scheduling is *scheduler-agnostic* as it works with any scheduler (*e.g.*, a FIFO queue, or a priority queue). The details of job placement (which physical machine runs each job) and job prioritization (which jobs run first) are up to the scheduler.

Whereas the Google approach plans a whole day's VCC at once, `DatacenterEnv` allows RL agents to plan the VCC one hour at a time. Furthermore, `DatacenterEnv` makes the following simplifying assumptions compared to the Google approach described above:

- `DatacenterEnv` does not provide any predictions of the next-day predicted total daily capacity, instead relying on the reward function to penalize an agent for failing to plan a VCC that meets 97% of the day's capacity.
- `DatacenterEnv` does not account for peak power consumption.
- `DatacenterEnv` uses a simple scheduler based on a priority queue, and jobs are randomly placed on any available machine. Machine constraints are not accounted for.

In DatacenterEnv, the data used to simulate a datacenter workload is subsampled from Cluster A from the Google Cluster Workload Traces May 2019 dataset [270]. The 47,276 jobs in our subsample include widely varying priorities, ranging from 0 (low priority) to 450 (high priority, latency-sensitive). We assume that our datacenter consists of 101 identical machines.

Training. We trained PPO, SAC, and deep deterministic policy gradient (DDPG) [163] RL algorithms on DatacenterEnv using PyTorch with RLLib's default hyperparameters for 60 episodes.

Representativeness, Generalizability, and Limitations. DatacenterEnv is loosely based on a Google data center from May 2019. As most datacenters do not disclose their exact job scheduling mechanisms and machine specifications, DatacenterEnv uses several simplifying assumptions. It uses a simple priority queue for scheduling jobs, and it assumes that there are no constraints for which jobs can be placed on which machine. However, DatacenterEnv does use a subsample of actual job traces from a Google datacenter in May 2019, which includes information for each job such as priority, duration, and compute usage. As DatacenterEnv is specifically based on the VCC framework used by Google's approach to carbon-aware datacenters, it does not reflect efforts by other cloud computing providers such as Microsoft [118] that may incorporate carbon emission estimates directly into the decision-making of their datacenter job schedulers.

Because Google only released datacenter job traces from May 2019, our ability to provide distribution shifts in DatacenterEnv over time is limited. We have thus chosen to only test shifts in the marginal carbon emissions rate, even though more realistic distribution shifts would also include changes in the statistics of datacenter jobs and in the capacities of the datacenter machines.

CogenEnv

System Model and Motivation. In the single-agent cogeneration environment, the agent controls a combined-cycle gas power plant with three gas turbines and one steam turbine. Gas power plants, similar to other conventional dispatchable generation types, suffer from loss of efficiency and degradation as a result of ramping energy generation up and down, thus necessitating the development of dispatch algorithms that can balance the tradeoff between fuel efficiency and ramp magnitude by anticipating future ramp needs while meeting demand in the highly constrained

decision space. Such algorithms are even more crucial during the ongoing energy transition, as large quantities of variable and intermittent solar and wind resources are added to the grid. The variability of these resources requires dispatchable generators to ramp more frequently to balance supply with demand, and thus, dispatch algorithms that have been deployed historically may be suboptimal if they do not consider these ramp needs. The problem of dispatchable power plant operation in the face of uncertain renewable generation is thus an important problem, as better algorithms for generator dispatch will ensure that the performance and fuel efficiency of thermal generators will not degrade as renewables penetration increases.

The foundation of the system model in `CogenEnv` is a neural network that maps ambient conditions and dispatch variables to plant fuel consumption and other variables related to plant operation and constraints. This model was developed in partnership with Beyond Limits and Enexsa. A summary of the plant model inputs and outputs is provided in Table 7.5. In the table, "GT" abbreviates "Gas Turbine" and "HRSG" abbreviates "heat recovery steam generator". In brief, a generator dispatch decision includes generator outputs and steam flows for each of the three gas turbines, along with auxiliary binary decisions concerning the state of an evaporative cooler and power augmentation mode for the gas turbine. The dispatch decision also includes a generator output, condenser steam flow, and cooling tower bay count for the steam turbine. This dispatch decision, when provided to the plant model in conjunction with ambient temperature, pressure, and relative humidity (together comprising entries 1 through 18 in Table 7.5), yield a number of outputs (entries 19 through 47) detailing the fuel consumption of each turbine unit (entries 23-31) and of the plant as a whole (entry 22), the electric power generation of the plant (entry 19), the plant steam production (entry 20), and a number of dynamic operating limits on the dispatch variables (entries 32-47).

Action Space. The action space for the single-agent environment is a vector $a(t) \in \mathbb{R}^{15}$ specifying a dispatch decision for the cogeneration plant, and specifically, specifying values for entries 4 through 18 of Tables 7.5.

Observation Space. As described in the main body of the paper, observations take the form

$$s(t) = (\tau, a(t-1), T_{t:t+k}, P_{t:t+k}, H_{t:t+k}, d^p_{t:t+k}, d^q_{t:t+k}, \pi^p_{t:t+k}, \pi^f_{t:t+k}).$$

where τ is a normalized time – we consider each episode to be a single day, broken into 15 minute intervals, so $\tau = t/96$ – and $a(t-1)$ is the previous action. $T_{t:t+k}$, $P_{t:t+k}, H_{t:t+k}, d^p_{t:t+k}, d^q_{t:t+k}, \pi^p_{t:t+k}$ and $\pi^f_{t:t+k}$ are vectors of the current and k steps of future forecasts of temperature, pressure, relative humidity, electricity demand, steam demand, electricity price, and fuel price, respectively. The units and limits of temperature, pressure, and relative humidity are as in entries 1 through 3 of Table 7.5, electricity demand has units MW, and steam demand is in klb/h. While we do not use electricity and fuel prices in the reward for this environment, we include them in the observation to allow for modification of the agent reward to incorporate financial incentives; gas price data was obtained from the Henry Hub Natural Gas Spot Price dataset (https://www.eia.gov/dnav/ng/hist/rngwhhdm.htm) and electricity price data was obtained from historical day-ahead prices at the Houston zone of the ERCOT grid (https://www.ercot.com/mp/data-products/data-product-details?id=NP4-180-ER).

Reward Function The reward for the agent is defined as

$$r(t) = -\left(r_f(a(t); T_t, P_t, H_t) + r_r(a(t); a(t-1)) + r_c(a(t); d^p_t, d^q_t)\right).$$

The term $r_f(a(t); T_t, P_t, H_t)$ is the generator fuel consumption in response to dispatch decision $a(t)$; this is exactly the total fuel consumption of the plant (entry 22 of Table 7.5) resulting from model inputs $(T_t, P_t, H_t, a(t))$. The term $r_r(a(t); a(t-1))$ is the cost of ramping electricity generation up or down. Defining $a_j(t)$ to be the jth entry of Table 7.5 (so j ranges from 4 to 18 to include all entries comprising the action space), the ramp cost is defined as

$$\begin{aligned} r_r(a(t); a(t-1)) = \beta \cdot (&|a_4(t) - a_4(t-1)| + |a_8(t) - a_8(t-1)| \\ &+ |a_{12}(t) - a_{12}(t-1)| + |a_{16}(t) - a_{16}(t-1)|). \end{aligned}$$

In our experiments, we set the penalty magnitude $\beta = 2$, following [58]. The third term, $r_c(a(t); d^p_t, d^q_t)$, penalizes two types of constraint violation: the first is supply-demand imbalance. Let $x(t) \in \mathbb{R}^{47}$ be a vector containing plant model inputs and outputs as in Table 7.5; then $x_{19}(t)$ is the total power output of the plant and $x_{20}(t)$ is the total steam output of the plant. The form of the penalty on supply-demand imbalance is as follows:

$$r^{\text{sd}}_c(a(t); d^p_t, d^q_t) = \gamma \cdot \left(\max\{0, d^p_t - x_{19}(t)\} + \max\{0, d^q_t - x_{20}(t)\}\right).$$

The second type of constraint violation penalized is that of the dynamic operating constraints. These are determined by the plant outputs in entries 32 through 47

of Table 7.5, and constrain dispatch variables to lie within certain intervals. This penalty takes the following form:

$$\begin{aligned} r_c^{\text{dyn}}(a(t); d_t^p, d_t^q) = \gamma \cdot \big(& \max\{0, x_4(t) - x_{32}(t)\} + \max\{0, x_{33}(t) - x_4(t)\} \\ & + \max\{0, x_8(t) - x_{34}(t)\} + \max\{0, x_{35}(t) - x_8(t)\} \\ & + \max\{0, x_{12}(t) - x_{36}(t) + \max\{0, x_{37}(t) - x_{12}(t)\} \\ & + \max\{0, x_7(t) - x_{38}(t)\} + \max\{0, x_{39}(t) - x_7(t)\} \\ & + \max\{0, x_{11}(t) - x_{40}(t)\} + \max\{0, x_{41}(t) - x_{11}(t)\} \\ & + \max\{0, x_{15}(t) - x_{42}(t)\} + \max\{0, x_{43}(t) - x_{15}(t)\} \\ & + \max\{0, x_{17}(t) - x_{44}(t)\} + \max\{0, x_{17}(t) - x_{45}(t)\} \\ & + \max\{0, x_{16}(t) - x_{46}(t)\} + \max\{0, x_{47}(t) - x_{16}(t)\}\big). \end{aligned}$$

In our experiments, we set the penalty magnitude parameter $\gamma = 1000$. The total constraint violation penalty is simply the sum of the supply-demand violation penalty $r_c^{\text{sd}}(a(t); d_t^p, d_t^q)$ and the dynamic operating constraint penalty $r_c^{\text{dyn}}(a(t); d_t^p, d_t^q)$.

Distribution Shift. In our distribution shift scenario, we consider the addition of 300 MW of wind energy onto the grid, which increases variability of net energy demand and changes the shape of load profiles. We obtain simulated wind speed profiles using the WIND Toolkit [131] at an altitude of 100m at $(39.970406, -128.77481)$ at 15 minute intervals, and transform these into electricity generation profiles using the power curve for an IEC Class 2 turbine, scaling to obtain a maximum generation level of 300 MW.

Multi-Agent Setting. In the multi-agent version of the environment, each of the four agents represents one of the blocks of the cogeneration plant: the first three control the three gas turbine blocks, and the fourth controls the steam generation block. Each agent observes the global observation, but controls only the variables relevant for its block — thus, agent 1 controls variables 4 through 7 in Table 7.5, agent 2 controls variables 8 through 11, agent 3 controls variables 12 through 15, and agent 4 controls variables 16 through 18. Each agent's reward only includes the terms relevant for its unit - thus, for instance, agent 1 pays fuel cost according to entry 25, ramp cost $\beta \cdot (|a_4(t) - a_4(t-1)|)$, and dynamic operating constraint penalty $\gamma \cdot \big(\max\{0, x_4(t) - x_{32}(t)\} + \max\{0, x_{33}(t) - x_4(t)\} + \max\{0, x_7(t) - x_{38}(t)\} + \max\{0, x_{39}(t) - x_7(t)\}\big)$. However, the supply-demand imbalance constraint penalty is shared equally amongst all agents: each pays $\frac{1}{4} r_c^{\text{sd}}(a(t); d_t^p, d_t^q)$.

Training. We train and test the performance of several algorithms on 250 days of data (ambient conditions, demands, and prices) between May 2021 and January 2022. For the testing environment, we use the scenario with 300 MW of wind generation. We performed two splits: 1) by algorithm (PPO vs. random actions), and 2) training data with out-of-distribution data (300 MW wind generation) vs. in-distribution data (no wind generation). We tested 3 learning rates for PPO and multi-agent PPO: 5e-6, 5e-5, 5e-4. For each learning rate, three different random seeds were tested, and the model with the best training performance was selected.

Representativeness, Generalizability, and Limitations `CogenEnv` is based on an actual combined-cycle power plant operated in the U.S.A. The general configuration has remained unchanged, but certain parameters such as generation capacity of units have been changed to protect the original data. More specifically, the general configuration of gas turbines, steam turbine, cooling tower, and binary setpoints are the same. Temperature and pressure settings of intermediate pressure and high pressure steams are also the same. However, some design parameters such as power and steam generation capacities of each unit and units' efficiencies has been scaled/shifted.

According to our collaborator Mehdi Hosseini at Beyond Limits: "Gas and steam turbines are generally modeled by simulating the Brayton and Rankine thermodynamic cycles. Besides the considered setpoints, some gas turbines might include extra minor setpoints like switching the duct burner on and off, which is considered always on in this model. The behavior of the steam turbine is more complex and interrelated with the structure of the condenser and cooling tower. In this simulation, a forced air cooling tower structure is employed, a typical method in combined cycle power plants. It's worth noting that using a different cooling system might influence the efficiency and generation limits of the steam turbine. In summary, the model accurately represents single or multi-gas turbine power plants, as well as combined-cycle power plants with a forced air cooling tower."

Finally, the distribution shift modeled in `CogenEnv` comes from changes in external renewable wind energy penetration, which causes greater need for ramping of fuel-based generators, leading to higher ramping costs. The is reflective of actual changes and challenges occurring in modern electricity grids, often referred to as the "duck curve problem" [48]. However, `CogenEnv` currently does not model other sources of distribution shift such as changing fuel prices.

BuildingEnv

Observation space. `BuildingEnv` considers a building with M indoor zones. The observation $s(t) \in \mathbb{R}^{M+4}$ is the concatenation of the zonal observations $T(t) \in \mathbb{R}^M$ and the environmental observations $s^{\text{env}}(t) \in \mathbb{R}^4$:

$$\begin{aligned} s(t) &= [T(t)^\top, s^{\text{env}}(t)^\top]^\top \\ T(t) &= [T_1(t), \ldots, T_M(t)]^\top \\ s^{\text{env}}(t) &= [T_{\text{E}}(t), T_{\text{G}}(t), Q^{\text{GHI}}(t), \bar{Q}^{\text{p}}(t)]^\top. \end{aligned}$$

We split the observation $s(t)$ into the two components to emphasize that $T(t)$ is affected by the agent's control actions, whereas $s^{\text{env}}(t)$ is the set of exogenous time-varying environmental variables that are unaffected by the agent's control actions. The descriptions for each of these variables can be found in Table 7.6.

Action space. The action space in `BuildingEnv` can be configured to operate in either continuous or discretized mode.

In the continuous mode, the action $a(t) = [a_1(t), \ldots, a_M(t)]^\top$ represents a set of controllable actions for building heat control, where $a_i(t)$ is the controlled heating supplied to zone i, normalized such that $a_i(t) \in [-1, 1]$.

In the discretized mode, the action space is quantized into a finite set of discrete values. Specifically, the action for each zone i is represented as an integer $a_i^{\text{discrete}}(t)$, where $a_i^{\text{discrete}}(t) \in [0, 200]$. This discrete representation is derived by scaling and translating the continuous action $a_i(t)$ from the range $[-1, 1]$ to the discrete range $[0, 200]$. After the environment selects a discrete action, it is remapped back to the continuous range of $[-1, 1]$ by

$$a_i(t) = \left(\frac{a_i^{\text{discrete}}(t)}{100} \right) - 1,$$

where the scaling and translation align the discrete action space $[0, 200]$ with the continuous action space $[-1, 1]$. This discretization allows for a finite, yet sufficiently granular, set of actions and enables the action space to adapt to environments where a more discrete control is required.

The resulting heat supply from the HVAC system to zone i at time t is given by

$$Q_i^{\text{h}}(t) = a_i w_i Q_i^{\text{h,max}},$$

where w_i is the efficiency coefficient for the HVAC system in zone i and $Q_i^{\mathrm{h,max}}$ is the maximum heat supply from the HVAC in zone i.

Given an action $a(t)$, the `BuildingEnv.step()` method simulates the next state via the physics-based state transition model described below.

System model. The building simulation builds upon the physics-principled environment in [318], which includes a reduced linear Resistance-Capacitance (RC) model for heat transfer with nonlinear residual modeling for occupants' activities and solar irradiance. The zonal thermal dynamics are given by

$$C_i \frac{dT_i}{dt} = \sum_{j \in \mathcal{N}(i)} \frac{T_j - T_i}{R_{i,j}} + Q_i^{\mathrm{h}} + Q_i^{\mathrm{s}} + Q_i^{\mathrm{p}}, \tag{7.5}$$

where $\mathcal{N}(i)$ is the set of zones neighboring zone i. That is,

$$\mathcal{N}(i) = \{j \in \{1, \ldots, M, \mathrm{G}, \mathrm{E}\} \mid j \neq i, \text{ zone } j \text{ shares a wall with zone } i\}.$$

The set $\mathcal{N}(i)$ may include G and E, if zone i is on the ground floor or connected to the outside environment, respectively. C_i is the thermal capacitance (in J/K), and $R_{i,j} = R_{j,i}$ is the thermal resistance (in K/W) between zones i, j. If zone i and j are not neighboring zones (*i.e.*, $j \notin \mathcal{N}(i)$), we set $R_{i,j} = R_{j,i} = +\infty$. Q_i^{h} is the controlled heating/cooling flow distributed to each zone as described above. Q_i^{s} and Q_i^{p} adhere to models outlined in the EnergyPlus documentation [71], which represent the heat accumulation in zone i from solar heat acquisition from windows and indoor human activities, respectively.

To capture the solar heat acquisition from windows, let α_{GHI} denote the coefficient determining the solar heat gain for windows, A_i^{win} be the window area for zone i (in m^2), and $Q^{\mathrm{GHI}}(t)$ be the heat gain from global horizontal irradiance (in $\mathrm{W/m}^2$). Then the accumulated solar heat from windows at time t can be calculated as

$$Q_i^{\mathrm{s}}(t) = \alpha_{\mathrm{GHI}} \cdot A_i^{\mathrm{win}} \cdot Q^{\mathrm{GHI}}(t).$$

To calculate heat gain originating from human activities at time t, $Q_i^{\mathrm{p}}(t)$, we consider N_i to symbolize the population in zone i, and $\bar{Q}^{\mathrm{p}}(t)$ signifies the discernible heat contributed by a single individual's activities. We assume that the number of people in each zone does not change over the course of the simulation, and we leave the task of modeling time-varying population for future work. Then,

$$Q_i^{\mathrm{p}}(t) = \bar{Q}^{\mathrm{p}}(t) \cdot N_i.$$

The computation of sensible heat per individual denoted as $\bar{Q}_p(t)$, is given by a polynomial function from the EnergyPlus documentation [71, p.1299],

$$\begin{aligned}\bar{Q}^{\mathrm{p}}(t) = c_1 + c_2 m_t + c_3 m_t^2 + c_4 \bar{T}(t) - c_5 m_t \bar{T}(t) + c_6 m_t^2 \bar{T}(t) - c_7 \bar{T}^2(t) \\ + c_8 m_t \bar{T}^2(t) - c_9 m_t^2 \bar{T}^2(t),\end{aligned}$$

where m_t signifies the population metabolic rate (in W) at time t, $\bar{T}(t) = \frac{1}{M}\sum_{i=1}^{M} T_i(t)$ is the average zone temperature, and $c_1, \ldots, c_9$ are constants that have been deduced by fitting sensible heat data under a variety of conditions. Both $Q_i^{\mathrm{s}}(t)$ and $Q_i^{\mathrm{p}}(t)$ fluctuate over time and embody heat emanations from the environment and occupants' activities that are beyond control.

The state evolution in (7.5), together with the definitions of $Q_i^{\mathrm{h}}, Q_i^{\mathrm{s}}, Q_i^{\mathrm{p}}$ can be written as

$$\dot{T}(t) = AT(t) + Bu(t) + Df(T(t)), \tag{7.6}$$

where $u(t) = [T_{\mathrm{G}}(t), T_{\mathrm{E}}(t), a_1(t), \ldots, a_M(t), Q^{\mathrm{GHI}}(t)]^\top$. Let the indicator variables $\mathbf{1}_{\mathrm{G},i} := \mathbf{1}[\mathrm{G} \in \mathcal{N}(i)]$ and $\mathbf{1}_{\mathrm{E},i} := \mathbf{1}[\mathrm{E} \in \mathcal{N}(i)]$ encode zone i's connectivity to the ground and the outside environment, respectively. Then the A and B matrices are

$$A = \begin{bmatrix} \sum_{j\in\mathcal{N}(i)} \frac{-1}{C_1 R_{1,j}} + \frac{c_p N_1}{MC_1} & \frac{1}{C_1 R_{1,2}} + \frac{c_p N_1}{MC_1} & \cdots & \frac{1}{C_1 R_{1,M}} + \frac{c_p N_1}{MC_1} \\ \frac{1}{C_2 R_{2,1}} + \frac{c_p N_2}{MC_2} & \sum_{j\in\mathcal{N}(i)} \frac{-1}{C_2 R_{2,j}} + \frac{c_p N_2}{MC_2} & \ddots & \vdots \\ \vdots & & \ddots & \frac{1}{C_{M-1} R_{M-1,M}} + \frac{c_p N_{M-1}}{MC_{M-1}} \\ \frac{1}{C_M R_{M,1}} + \frac{c_p N_M}{MC_M} & \cdots & \cdots & \sum_{j\in\mathcal{N}(i)} \frac{-1}{C_M R_{M,j}} + \frac{c_p N_M}{MC_M} \end{bmatrix},$$

$$B = \begin{bmatrix} \frac{\mathbf{1}_{\mathrm{G},1}}{C_1 R_{\mathrm{G},1}} & \frac{\mathbf{1}_{\mathrm{E},1}}{C_1 R_{\mathrm{E},1}} & \frac{w_1 Q_1^{\mathrm{h,max}}}{C_1} & 0 & \cdots & 0 & \frac{\alpha_{\mathrm{GHI}} A_1^{\mathrm{win}}}{C_1} \\ \frac{\mathbf{1}_{\mathrm{G},2}}{C_2 R_{\mathrm{G},2}} & \frac{\mathbf{1}_{\mathrm{E},2}}{C_2 R_{\mathrm{E},2}} & 0 & \frac{w_2 Q_2^{\mathrm{h,max}}}{C_2} & \cdots & 0 & \frac{\alpha_{\mathrm{GHI}} A_2^{\mathrm{win}}}{C_2} \\ \vdots & \vdots & \vdots & & \ddots & & \vdots \\ \frac{\mathbf{1}_{\mathrm{G},M}}{C_M R_{\mathrm{G},M}} & \frac{\mathbf{1}_{\mathrm{E},M}}{C_M R_{\mathrm{E},M}} & 0 & 0 & \cdots & \frac{w_M Q_M^{\mathrm{h,max}}}{C_M} & \frac{\alpha_{\mathrm{GHI}} A_M^{\mathrm{win}}}{C_M} \end{bmatrix}.$$

The vector D in Eq (7.6) is defined as $D = \left[\frac{N_1}{C_1}, \frac{N_2}{C_2}, \cdots, \frac{N_M}{C_M}\right]^\top$, and the nonlinear function for sensible heat calculation is

$$\begin{aligned} f(T(t)) &:= \bar{Q}^{\mathrm{p}}(t) - c_4 \bar{T}(t) \\ &= c_1 + c_2 m_t + c_3 m_t^2 - c_5 m_t \bar{T}(t) + c_6 m_t^2 \bar{T}(t) - c_7 \bar{T}^2(t) + c_8 m_t \bar{T}^2(t) - c_9 m_t^2 \bar{T}^2(t). \end{aligned}$$

We convert the continuous-time system model into a discrete-time model,

$$T[k+1] = \bar{A}T[k] + \bar{B}u[k] + D\bar{f}[k] \tag{7.7}$$

where the discrete-time system matrices $\bar{A}, \bar{B}, \bar{f}$ are obtained from the continuous-time model parameters $A, B, f(T(t))$ using zero-order hold method with discretization time ΔT. In particular, $\bar{A} = e^{A\Delta T}$, $\bar{B} = A^{-1}(\bar{A} - I)B$, and $\bar{f}[k] = \int_{k\Delta T}^{(k+1)\Delta T} f(T(\tau))\,\mathrm{d}\tau$.

Reward function. The main objective of building control is to reduce energy consumption while keeping the temperature within a given comfort range. Therefore, the reward function penalizes both temperature deviations and HVAC energy consumption:

$$r(t) = -(1-\beta)\,\|a(t)\|_p - \beta\|T^{\text{target}}(t) - T(t)\|_p\,,$$

where $T^{\text{target}}(t) = [T_1^{\text{target}}(t), \ldots, T_M^{\text{target}}(t)]^\top$ are the target temperatures. The parameters β and p are user-customizable scalars, where β trades off between the energy consumption and temperature deviation penalties, and p determines the norm used. An important future direction is to incorporate CO_2 emissions into consideration in the default reward function for building control environment.

Building and weather types for simulation. The prototype buildings included in `BuildingEnv` are derived from the Department of Energy (DOE) Commercial Reference Building Models. The models include 16 commercial building types in 19 locations. Users can download all models at https://www.energycodes.gov/prototype-building-models. Since `BuildingEnv` is compatible with EnergyPlus, users could also create their own model in the EnergyPlus editor and load the generated table file into the `BuildingEnv`.

- *Available building types*: ApartmentHighRise, ApartmentMidRise, Hospital, HotelLarge, HotelSmall, OfficeLarge, OfficeMedium, OfficeSmall, OutPatientHealthCare, RestaurantFastFood, RestaurantSitDown, RetailStandalone, RetailStripmall, SchoolPrimary, SchoolSecondar, Warehouse.
- *Available cities and weather types*: Ho Chi Minh City (Extremely Hot Humid), Dubai (Extremely Hot Dry), Honolulu (Very Hot Humid), New Delhi (Very Hot Dry), Tampa (Hot Humid), Tucson (Hot Dry), Atlanta (Warm Humid), El Paso (Warm Dry), San Diego (Warm Marine), New York (Mixed Humid), Albuquerque (Mixed Dry), Seattle (Mixed Marine), Buffalo (Cool Humid), Denver (Cool Dry), Port Angeles (Cool Marine), Rochester (Cold Humid), Great Falls (Cold Dry), International Falls (Very Cold), Fairbanks (Subarctic/Arctic).

Code:

```
from sustaingym.envs.building import BuildingEnv, ParameterGenerator
params = ParameterGenerator(
    building='OfficeLarge', weather='Warm_Marine', location='SanDiego')
env = BuildingEnv(params)
```

Output:

```
###############All Zones from Ground############
BASEMENT  [Zone index]:  0
DATACENTER_BASEMENT_ZN_6  [Zone index]:  1
CORE_BOTTOM  [Zone index]:  2
PERIMETER_BOT_ZN_3  [Zone index]:  3
PERIMETER_BOT_ZN_2  [Zone index]:  4
PERIMETER_BOT_ZN_1  [Zone index]:  5
PERIMETER_BOT_ZN_4  [Zone index]:  6
DATACENTER_BOT_ZN_6  [Zone index]:  7
GROUNDFLOOR_PLENUM  [Zone index]:  8
CORE_MID  [Zone index]:  9
PERIMETER_MID_ZN_3  [Zone index]:  10
PERIMETER_MID_ZN_2  [Zone index]:  11
PERIMETER_MID_ZN_1  [Zone index]:  12
PERIMETER_MID_ZN_4  [Zone index]:  13
DATACENTER_MID_ZN_6  [Zone index]:  14
MIDFLOOR_PLENUM  [Zone index]:  15
CORE_TOP  [Zone index]:  16
PERIMETER_TOP_ZN_3  [Zone index]:  17
PERIMETER_TOP_ZN_2  [Zone index]:  18
PERIMETER_TOP_ZN_1  [Zone index]:  19
PERIMETER_TOP_ZN_4  [Zone index]:  20
DATACENTER_TOP_ZN_6  [Zone index]:  21
TOPFLOOR_PLENUM  [Zone index]:  22
################################################
```

Figure 7.8: Importing OfficeLarge in `BuildingEnv`.

Example. A practical usage example is demonstrated in Figure 7.8. The intent of this code excerpt is to imitate an "OfficeLarge" type structure in San Diego, with "Warm Marine" weather conditions, employing arbitrarily selected actions. Upon the creation of a building model, comprehensive zone information is then displayed as shown in Figure 7.8. At each control time step, the RL agent observes the present state $s(t)$ and produces a corresponding action $a(t)$. The environment `BuildingEnv`, in turn, incorporates this action and integrates it into the building state-space model to project the subsequent state $s(t+1)$ for the approaching time step. Each episode runs for 1 day, with 5-minute time intervals ($H = 288$, $\tau = 5$ minutes). The agent controls the supplied heat flow to each zone and is rewarded for maintaining the desired temperature at the minimum electricity usage. Figure 7.9 visualizes the indoor temperature in different zones of OfficeLarge for 1 day without control, initialized at $13.2°C$.

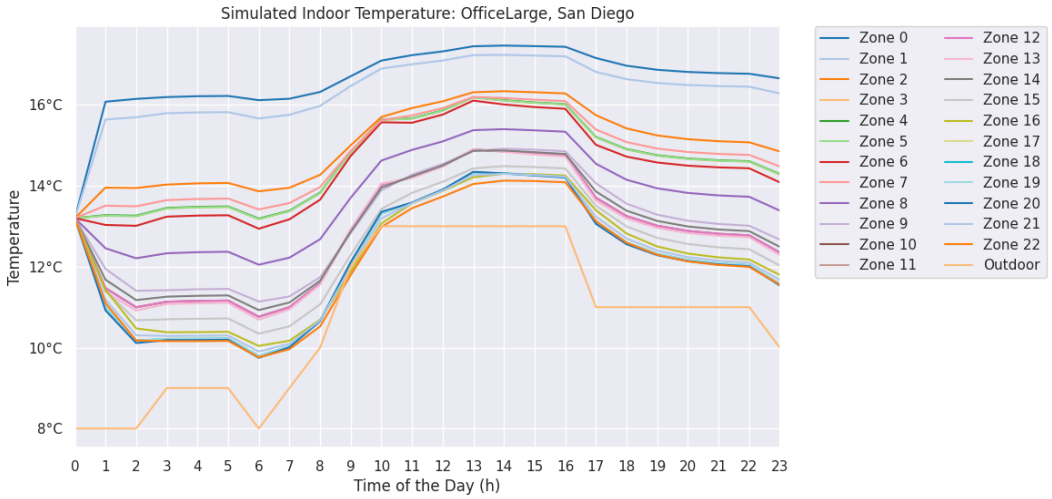


Figure 7.9: Simulated indoor temperature in different zones of OfficeLarge for 1 day without control.

Model predictive control (MPC). As a baseline non-RL algorithm, we consider a model predictive control (MPC) controller similar to the `EVChargingEnv` environment. At every time step t, with a lookahead window w, the MPC controller solves the following optimization problem,

$$\min_{a(t),\ldots,a(t+w-1)} \quad \sum_{k=t}^{t+w-1} (1-\beta)\,\|a(k)\|_2 + \beta\|T^{\text{target}}(k) - T(k)\|_2 \tag{7.8a}$$

$$\text{s.t.} \quad -1 \preceq a_i(k) \preceq 1\,, \forall i \in \{0,\ldots,M\}, \forall k \in \{t,\ldots,t+w-1\} \tag{7.8b}$$

$$\text{building dynamics in (7.7)}\,, \tag{7.8c}$$

and implements action $a^\star(t)$. MPC method has complete model knowledge and perfect predictions about building occupancy $[N_1, N_2, \ldots, N_M]$, the ground and environmental temperature $T_{\text{G}}(k), T_{\text{E}}(k)$, solar irradiance $Q^{\text{GHI}}(k)$ for $k = t, t+1, \ldots, t+w-1$. (7.8b) ensures that the control signals are feasible, and (7.8c) follows the building physics model in (7.7) which uses a zero-order hold discretization method to derive from the continuous-time model with discretization interval as $\Delta T = 1/12$ hour.

Multiagent Environment. The multiagent setting pairs each zone with one agent, so that each agent i is only responsible for action $a_i(t)$. Currently, we let every agent observe the complete building state space, and each agent receives the same global reward. For future directions, we plan on implementing a separate local state space

for each agent and also imposing a total power constraint (*e.g.*, $\|a(t)\|_1$ must be bounded by some limit).

Training We trained Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and Advantage Actor Critic (A2C) [177] Reinforcement Learning algorithms on the `BuildingEnv` environment using PyTorch and StableBaselines3. These algorithms were applied to an "OfficeSmall" type building situated in a "Hot Dry" climate, specifically in Tucson. The training and testing periods were set in the winter (January 2003) and summer (July 2004) sections, respectively. The models were trained during both the winter and summer sections, but testing was conducted solely in the winter section to assess the impact of distribution shifts. We experimented with two learning rates for each of PPO, SAC, and A2C: 3e-4 and 3e-5 for PPO and SAC, and 7e-4 and 3e-4 for A2C. The model demonstrating the best performance in training was subsequently selected for further analysis.

Representativeness and Generalizability `BuildingEnv` is designed to ensure broad generalizability across diverse building scenarios. We have integrated an extensive collection of standard prototype building models coupled with a variety of weather conditions, encompassing the majority of Reference Building types. These models are based on the Department of Energy (DOE)'s Commercial Reference Building Models, offering a robust foundation for simulation. Furthermore, the environment is highly customizable. Users have the flexibility to adjust parameters such as the wall material, the solar heat gain coefficient of windows, and the ground temperature specific to the building's location. For researchers and developers seeking even more specificity, there is also an option to define custom building models using EnergyPlus. Users can then input the resulting output input data file (IDF), along with the corresponding EPW weather file, into `BuildingEnv`, providing a custom simulation experience.

Limitations Beyond the modeled weather variations, occupancy dynamics also introduce distribution shifts in building energy management. While `Building Env` provides mechanisms to simulate occupants in different zones with set activity schedules, truly replicating the unpredictability of human behavior remains a complex endeavor. For instance, we can configure a room to have four individuals engaged in seated work from 1pm to 4pm, followed by three individuals running from 4pm to 5pm, with the room being vacant after 5pm. Despite these features, there are

certain distribution shifts that `BuildingEnv` does not currently address, such as equipment failures, external environmental impacts like nearby construction or urban heat island effects, and the nuances of building aging. However, it is worth noting that the modular design of `BuildingEnv` provides a foundation that is conducive to future enhancements and adaptations, allowing for the incorporation of additional sources of distribution shift as our understanding of building dynamics evolves.

Table 7.3: `ElectricityMarketEnv` parameters. The k-th entry of a vector $\mathbf{g}_t$ is denoted $g_{k,t}$.

Param.	Domain	Unit	Description
t	$[1, \ldots, T]$	5 min	index of real-time interval (usually 5-15 min)
T	$\mathbb{Z}_+$		number of periods in each epoch, typically 1 day for 5 min intervals
h	$\{0, 1, \ldots, T\}$		length of the lookahead horizon
τ	$[t, \ldots, t+h]$	5 min	index of the interval in each multi-interval lookahead subproblem
N	$\mathbb{Z}_+$		number of nodes in network
M	$\mathbb{Z}_+$		number of lines in the network
N_G	$\mathbb{Z}_+$		number of generators
N_D	$\mathbb{Z}_+$		number of loads
N_B	$\mathbb{Z}_+$		number of batteries
i	$[1, \ldots, N]$		index of node
j	$[1, \ldots, M]$		index of line
k	$[1, \ldots, N_G, N_D, N_B]$		index of generator, load, or battery
B	$\mathbb{R}_+^{M \times M}$	Ω^{-1}	network susceptance matrix
C	$\{-1, 0, 1\}^{N \times M}$		node-line network incidence matrix
Σ	$\{-1, 0, 1\}^{N \times (N_D + N_G + 2N_B)}$		participant-to-node mapping matrix
H	$\mathbb{R}^{M \times N}$		generation shift factor matrix
$\mathbf{f}^{\max}$	$\mathbb{R}_+^M$	MVA	maximum power flow along each line
$\hat{\mathbf{d}}_t$	$\mathbb{R}^{N_D}$	MW	vector of predicted (average) load in interval t
$\underline{\mathbf{g}}, \overline{\mathbf{g}}$	$\mathbb{R}_+^{N_G}$	MW	min/max generation limits (assumed to be time invariant)
$\overline{\mathbf{b}^{\mathrm{c}}}, \overline{\mathbf{b}^{\mathrm{d}}}$	$\mathbb{R}_+^{N_B}$	MW	max charge/discharge limits (assumed to be time invariant)
$\underline{\mathbf{x}}, \overline{\mathbf{x}}$	$\mathbb{R}_+^{N_B}$	MWh	min/max state of charge limits (assumed to be time invariant)
$\mathbf{x}^{\mathrm{f}}$	$\mathbb{R}_+^{N_B}$	MWh	final state of charge required at interval T
$\mathbf{x}^0$	$\mathbb{R}_+^{N_B}$	MWh	initial state of charge at beginning of epoch
$\eta_k^{\mathrm{c}}, \eta_k^{\mathrm{d}}$	$(0, 1]$		charge and discharge efficiency of battery k
$c_{k,t}^{\mathrm{g}}$	$\mathbb{R}_+$	\$/MWh	marginal cost of generator k in interval t
$c_{k,t}^{\mathrm{b,c}}$	$\mathbb{R}_+$	\$/MWh	marginal cost of battery k charging in interval t
$c_{k,t}^{\mathrm{b,d}}$	$\mathbb{R}_+$	\$/MWh	marginal cost of battery k discharging in interval t

Table 7.4: Definitions of problem variables. Variables are optimized in the economic dispatch problem.

Variable	Domain	Unit	Description
$\mathbf{p}_t$	$\mathbb{R}^N$	MW	vector of nodal power injections in interval t
$\mathbf{d}_t$	$\mathbb{R}^{N_D}$	MW	vector of load power injections in interval t
$\mathbf{g}_t$	$\mathbb{R}^{N_G}$	MW	vector of generator power injections in interval t
$\mathbf{b}_t^{\mathrm{c}}$	$\mathbb{R}_+^{N_B}$	MW	vector of battery charging power injections in interval t
$\mathbf{b}_t^{\mathrm{d}}$	$\mathbb{R}_+^{N_B}$	MW	vector of battery discharging power injections in interval t
$\mathbf{x}_t$	$\mathbb{R}_+^{N_B}$	MWh	vector of battery states of charge in interval t
λ_t	$\mathbb{R}$		Lagrange multiplier corresponding to the power balance constraint $\mathbf{1}^\top \mathbf{p}_t = 0$
$\boldsymbol{\mu}_t^{\pm}$	$\mathbb{R}^M$		Lagrange multiplier vectors corresponding to upper/lower line limit constraints
$\boldsymbol{\pi}_t$	$\mathbb{R}^N$	\$/MWh	market-clearing nodal price vector for interval t

Table 7.5: Summary of input and output variables for `CogenEnv` plant model

Num.	Variable Description	Variable Group	Type	Feasible Region	Unit
1	Air Temperature	Ambient Conditions	Input	$[32, 115]$	°F
2	Air Pressure	Ambient Conditions	Input	$[14, 15]$	PSIA
3	Air Relative Humidity	Ambient Conditions	Input	$[0, 1]$	-
4	GT1 Generator Output	Turbine Block 1	Input	$[41.64, 168.3]$	MW
5	GT1 Evaporative Cooler Switch	Turbine Block 1	Input	$\{0, 1\}$	-
6	GT1 Power Augmentation Switch	Turbine Block 1	Input	$\{0, 1\}$	-
7	GT1 Steam Flow	Turbine Block 1	Input	$[403.2, 819.6]$	klb/h
8	GT2 Generator Output	Turbine Block 2	Input	$[41.49, 168.4]$	MW
9	GT2 Evaporative Cooler Switch	Turbine Block 2	Input	$\{0, 1\}$	-
10	GT2 Power Augmentation Switch	Turbine Block 2	Input	$\{0, 1\}$	-
11	GT2 Steam Flow	Turbine Block 2	Input	$[396.7, 817.4]$	klb/h
12	GT3 Generator Output	Turbine Block 3	Input	$[46.46, 172.4]$	MW
13	GT3 Evaporative Cooler Switch	Turbine Block 3	Input	$\{0, 1\}$	-
14	GT3 Power Augmentation Switch	Turbine Block 3	Input	$\{0, 1\}$	-
15	GT3 Steam Flow	Turbine Block 3	Input	$[439.0, 870.3]$	klb/h
16	Steam Generator Output	Steam Turbine	Input	$[25.65, 83.54]$	MW
17	Steam Flow through Condenser	Steam Turbine	Input	$[-1218, -318.1]$	klb/h
18	Number of Cooling Tower Bays	Steam Turbine	Input	$\{1, \ldots, 12\}$	-
19	Net Electric Power Output	Plant	Output	-	MW
20	Net Steam Export	Plant	Output	-	klb/h
21	Auxiliary Power Consumption	Plant	Output	-	MW
22	Total Fuel Consumption	Plant	Output	-	klb/h
23	GT1 Fuel Flow	Turbine Block 1	Output	-	klb/h
24	HRSG1 Fuel Flow	Turbine Block 1	Output	-	klb/h
25	Block 1 Total Fuel Consumption	Turbine Block 1	Output	-	klb/h
26	GT2 Fuel Flow	Turbine Block 2	Output	-	klb/h
27	HRSG2 Fuel Flow	Turbine Block 2	Output	-	klb/h
28	Block 2 Total Fuel Consumption	Turbine Block 2	Output	-	klb/h
29	GT3 Fuel Flow	Turbine Block 3	Output	-	klb/h
30	HRSG3 Fuel Flow	Turbine Block 3	Output	-	klb/h
31	Block 3 Total Fuel Consumption	Turbine Block 3	Output	-	klb/h
32	GT1 Min Power	Turbine Block 1	Output	-	MW
33	GT1 Max Power	Turbine Block 1	Output	-	MW
34	GT2 Min Power	Turbine Block 2	Output	-	MW
35	GT2 Max Power	Turbine Block 2	Output	-	MW
36	GT3 Min Power	Turbine Block 3	Output	-	MW
37	GT3 Max Power	Turbine Block 3	Output	-	MW
38	GT1 Min Steam	Turbine Block 1	Output	-	klb/h
39	GT1 Max Steam	Turbine Block 1	Output	-	klb/h
40	GT2 Min Steam	Turbine Block 2	Output	-	klb/h
41	GT2 Max Steam	Turbine Block 2	Output	-	klb/h
42	GT3 Min Steam	Turbine Block 3	Output	-	klb/h
43	GT3 Max Steam	Turbine Block 3	Output	-	klb/h
44	Steam Let-Down Flow Min Limit	Steam Turbine	Output	-	klb/h
45	Steam Let-Down Flow Max Limit	Steam Turbine	Output	-	klb/h
46	Steam Turbine Min Power	Steam Turbine	Output	-	MW
47	Steam Turbine Max Power	Steam Turbine	Output	-	MW

Table 7.6: Definitions of `BuildingEnv` variables.

Variable	Domain	Unit	Description
$T_i(t)$	$\mathbb{R}$	°C	temperature in zone i at time t
$T_{\mathrm{G}}(t)$	$\mathbb{R}$	°C	ground temperature at time t
$T_{\mathrm{E}}(t)$	$\mathbb{R}$	°C	outdoor/ambient temperature at time t
$Q^{\mathrm{GHI}}(t)$	$\mathbb{R}_+$	W/m^2	heat gain from solar irradiance per square meter at time t ("GHI" is an acronym for "global horizontal irradiance")
$Q_i^{\mathrm{s}}(t)$	$\mathbb{R}_+$	W	heat acquisition from solar irradiance in zone i at time t
$Q_i^{\mathrm{p}}(t)$	$\mathbb{R}_+$	W	heat acquisition from occupants' activities in zone i at time t
$Q_i^{\mathrm{h}}(t)$	$\mathbb{R}$	W	heat acquisition from HVAC in zone i at time t
$N_i(t)$	$\mathbb{N}$		number of occupants in zone i at time t
A_i^{win}	$\mathbb{R}_+$	m^2	window area for zone i
$Q_i^{\mathrm{h,max}}$	$\mathbb{R}_+$	W	maximum heat acquisition from HVAC in zone i
$a_i(t)$	$[-1, 1]$		action value, normalized heat acquisition from HVAC in zone i at time t (+ for heating, − for cooling)
w_i	$[0, 1]$		efficiency coefficient for HVAC in zone i

Chapter 8

DISTRIBUTIONALLY ROBUST COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING VIA ROBUST VALUE FACTORIZATION

Motivated by the shortcomings of existing RL and multi-agent RL algorithms seen on SustainGym from the previous chapter, this chapter investigates the problem of learning robust policies for cooperative multi-agent reinforcement learning (MARL) under distribution shifts. In particular, we focus on value-factorization methods for cooperative MARL, which are a popular class of algorithms that learn a centralized value function that can be factorized into individual value functions for each agent. These methods typically rely on the individual-global-maximum (IGM) principle, which states that the optimal joint action can be recovered by taking the greedy action with respect to each agent's individual value function. However, in real-world settings, environmental uncertainties can lead to distribution shifts that violate the IGM principle, resulting in suboptimal performance. To address this issue, we introduce the concept of Distributionally Robust IGM (DrIGM), which requires that each agent's robust greedy action aligns with the robust team-optimal joint action. We show that DrIGM holds for a novel definition of robust individual action values, which is compatible with decentralized greedy execution and yields a provable robustness guarantee for the whole system. Empirically, on high-fidelity SustainGym simulators and a StarCraft game environment, our methods consistently improve out-of-distribution performance.

This chapter is primarily based on the following paper:

[209] Chengrui Qu, Christopher Yeh, Kishan Panaganti, Eric Mazumdar, and Adam Wierman. 2026. Distributionally robust cooperative multi-agent reinforcement learning with value factorization. In *The Fourteenth International Conference on Learning Representations*. (Apr. 2026). https://openreview.net/forum?id=2T3L0pqI0O.

8.1 Introduction

Multi-agent reinforcement learning (MARL) is a popular framework for studying how multiple agents compete or cooperate in complex environments such as video game playing [276], economic policy design [325], wireless network communications [211], and power grid control [94], among others. In this work, we focus on the *cooperative* MARL setting, where each agent can only observe its local history, and

agents must collaborate to achieve a joint goal. To address partial observability and reduce real-time communication costs, a widely used paradigm is centralized training with decentralized execution (CTDE) [193]. During training, agents may aggregate global information, coordinate credit assignment, and learn a team structure; at deployment, each agent must act myopically based on its own local history.

The CTDE paradigm is typically realized through value factorization methods (e.g., VDN [259], QMIX [217], QTRAN [251]). A key concept that underpins the success of these methods is the individual-global-maximum (IGM) principle [251], which aligns each agent's greedy action with the team-optimal joint action via a suitable value factorization. However, most examples where the success of this principle is demonstrated are in virtual tasks (e.g., games [277] and grid worlds [150]). It remains unclear whether this principle maintains its reliability in real-world domains, where modeling is imperfect and execution is noisy.

In practice, a major obstacle facing cooperative MARL is environmental uncertainty [242]: team performance can drop sharply when the deployed environment deviates from the training environment due to model mismatch, system noise, and the sim-to-real gap [320, 16]. While environmental uncertainty presents challenges in single-agent RL settings, it is a more significant hurdle in cooperative MARL, where partial observability and inter-agent coupling can cause small mismatches to cascade into coordination failures [50, 107].

In single-agent RL, uncertainty in the environment is commonly addressed by distributionally robust RL (DR-RL) techniques [290, 265, 189, 196, 241] which seek policies that perform well under adversarial perturbations of a nominal environment model. Single-agent DR-RL is well-explored; however extending DR-RL to the cooperative MARL setting is fundamentally more challenging. In particular, each agent acts on a local history yet shares a team reward coupled with teammates' actions, making it nontrivial to define individual robust Q-functions that both evaluate worst-case outcomes and remain compatible with IGM for decentralized greedy execution. Reward engineering can help empirically, but only a narrow class of shaping functions can provably preserve optimality [86], even in the single-agent setting. Thus, *we seek a principled route to distributional robustness for cooperative MARL that remains compatible with decentralized greedy execution.*

Contributions. In this chapter, we introduce a family of distributionally robust cooperative MARL algorithms for the CTDE setting. Our central technique is **D**istributionally **r**obust **IGM** (DrIGM), a robustness principle that requires each

agent's robust greedy action to coincide with the robust team-optimal joint action, thereby preserving decentralized greedy execution.

We first show, via a concrete counterexample, that naïvely adopting individual robust action-value functions from single-agent DR-RL, where each agent considers its own worst case, does *not* guarantee decentralized alignment (IGM) in the cooperative multi-agent setting. We then provide a sufficient condition under which DrIGM holds: when individual robust action-value functions are defined with respect to the worst-case joint action-value function, DrIGM is guaranteed.

Next, we derive DrIGM-compliant robust variants of existing value factorization architectures (VDN, QMIX, and QTRAN), by training on robust Q-targets while retaining the CTDE information structure. The resulting methods are scalable, easy to implement on top of existing codebases, and maintain robustness at execution without requiring bespoke individual robust value design.

Finally, we evaluate our DrIGM-based algorithms on a realistic simulation of an HVAC control task in SustainGym [310], as well as on SMAC, a StarCraft II-based multi-agent game-playing environment [228]. Across out-of-distribution settings, our methods outperform non-robust value factorization baselines and a recent robust cooperative MARL baseline, consistently mitigating sim-to-real degradation on operational metrics.

Brief discussion of related work. Robustness in cooperative MARL has been studied along several axes: adversarial or heterogeneous teammates [153, 125, 157], state/observation and communication perturbations [100, 317], risk-sensitive (tail-aware) objectives under a fixed model [239], and explicit model uncertainty [143, 320, 319, 165]. Most of the works on model uncertainty adopt a distributionally robust optimization viewpoint and targets Nash solutions with provable algorithms, often assuming full observability or individual rewards [319, 127, 170, 35, 242, 165]. In this work, we focus on the cooperative CTDE regime with partial observability and a single team reward, providing a systematic framework for robustness to model uncertainty without real-time communication. For an extended discussion of related works, see Section 8.7.

8.2 Background and Problem Formulation

Notation. For a set $\mathcal{X}$, $|\mathcal{X}|$ denotes its *cardinality* and $\Delta(\mathcal{X})$ the probability simplex over $\mathcal{X}$. We write $\prod_i \mathcal{X}_i$ for the Cartesian product. For $N \in \mathbb{N}$, we let

$[N] := \{1, \ldots, N\}$. Let $[x]_+ := \max\{0, x\}$.

Cooperative Dec-POMDPs. A cooperative multi-agent task with N agents is modeled as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP):

$$G = (\mathcal{S}, \{\mathcal{A}_i\}_{i=1}^N, P, r, \{\mathcal{O}_i\}_{i=1}^N, \{\sigma_i\}_{i=1}^N, \gamma),$$

with joint action space $\mathcal{A} := \prod_{i\in[N]} \mathcal{A}_i$. At time t, each agent i obtains an individual observation $o_i^t := \sigma_i(s^t)$ from its observation space $\mathcal{O}_i$, chooses an action $a_i^t \in \mathcal{A}_i$, a bounded joint reward $r(s^t, \mathbf{a}^t)$ is received, where $\mathbf{a}^t := (a_1^t, \ldots, a_N^t) \in \mathcal{A}$ is the joint action, and then the state evolves via $s^{t+1} \sim P(\cdot \mid s^t, \mathbf{a}^t)$. Here we assume the joint observation $(o_1, \ldots, o_N)$ can recover the full state.[1] Each agent i acts using a history-based policy $\pi_i(\cdot \mid h_i^t)$ with $h_i^t := (o_i^0, a_i^0, \ldots, o_i^{t-1}, a_i^{t-1}, o_i^t)$; the joint policy is $\pi = \langle \pi_1, \ldots, \pi_N \rangle$. We use $\mathcal{H}_i^t$ to denote the space of possible histories for agent i up to time t. In the following sections, we will omit the superscript t to avoid notational clutter. The joint action-observation history is denoted $\mathbf{h} \in \mathcal{H} := \prod_{i\in[N]} \mathcal{H}_i$. Given the current state s, the joint history $\mathbf{h}$ and the joint action $\mathbf{a}$, we denote the joint action-value function under policy π by $Q_{\text{tot}}^{P,\pi}(\mathbf{h}, \mathbf{a})$, which can be reduced to $|\mathcal{S} \times \mathcal{A}|$ dimension as we assume the joint observation can recover the full state. We use $Q_{\text{tot}}^{P}(\mathbf{h}, \mathbf{a}) := \max_\pi Q_{\text{tot}}^{P,\pi}(\mathbf{h}, \mathbf{a})$ to denote the optimal joint action value.

CTDE. Centralized training with decentralized execution (CTDE) leverages global information during learning while executing individual policies from individual histories. A common CTDE approach is value factorization, which learns an optimal joint action-value $Q_{\text{tot}}^{P}(\mathbf{h}, \mathbf{a})$ and individual action-value functions $[Q_i^P(h_i, a_i)]_{i\in[N]}$ that satisfy the following *individual-global-max* (IGM) principle [251].

Definition 7 (IGM)**.** *We say that individual action-value functions* $[Q_i^P : \mathcal{H}_i \times \mathcal{A}_i \to \mathbb{R}]_{i\in[N]}$ *satisfy the* individual-global-max (IGM) principle *for an optimal joint action-value function* $Q_{\text{tot}}^P : \mathcal{H} \times \mathcal{A} \to \mathbb{R}$ *under joint history* $\mathbf{h} = (h_1, \ldots, h_N) \in \mathcal{H}$ *if*

$$\left(\arg\max_{a_1} Q_1^P(h_1, a_1), \ldots, \arg\max_{a_N} Q_N^P(h_N, a_N)\right) \subseteq \arg\max_{\mathbf{a}} Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}).$$

IGM ensures that greedy individual actions are jointly optimal w.r.t. Q_{tot}^P, enabling decentralized execution without test-time communication.

[1] Formally, we assume that $\sigma = (\sigma_1(\cdot), \ldots, \sigma_N(\cdot)) : \mathcal{S} \to \prod_{i\in[N]} \mathcal{O}_i$ is injective. Equivalently, there exists a (deterministic) decoding map $g : \prod_{i\in[N]} \mathcal{O}_i \to \mathcal{S}$ such that $g(\sigma(s)) = s$ for all $s \in \mathcal{S}$.

Robust Dec-POMDPs. To capture model error and deployment shift, we consider an *uncertainty set* $\mathcal{P}$ of environment models around a nominal P^0. In Dec-POMDPs, we work with a history-based view: let $P_{\mathbf{h},\mathbf{a}}(\cdot)$ denote the transition kernel over next joint histories $\mathbf{h}'$, given the current joint history $\mathbf{h}$ and the joint action $\mathbf{a}$. We assume a *history-action rectangular* uncertainty set.

$$\mathcal{P} = \prod_{(\mathbf{h},\mathbf{a})\in\mathcal{H}\times\mathcal{A}} \mathcal{P}_{\mathbf{h},\mathbf{a}}, \quad \mathcal{P}_{\mathbf{h},\mathbf{a}} \subseteq \Delta(\mathcal{H}), \tag{8.1}$$

e.g., balls around $P^0_{\mathbf{h},\mathbf{a}}$ under a probability metric with radius $\rho > 0$. This rectangularity assumption is widely adopted in DR-RL literature [35, 242, 170], and ensures that a robust policy exists.

Given a function $Q : \mathcal{H} \times \mathcal{A} \to \mathbb{R}$, define the *robust Bellman operator* $\mathcal{T}$ as

$$(\mathcal{T}Q)(\mathbf{h},\mathbf{a}) := r(s,\mathbf{a}) + \gamma \inf_{P_{\mathbf{h},\mathbf{a}}\in\mathcal{P}_{\mathbf{h},\mathbf{a}}} \mathbb{E}_{\mathbf{h}'\sim P_{\mathbf{h},\mathbf{a}}}\left[\max_{\mathbf{a}'\in\mathcal{A}} Q(\mathbf{h}',\mathbf{a}')\right]. \tag{8.2}$$

Under standard assumptions (bounded rewards, $\gamma \in (0,1)$) and rectangularity in eq. (8.1), $\mathcal{T}$ is a γ-contraction on the space of bounded Q, so it has a unique fixed point $Q^{\mathcal{P}}_{\text{tot}}$ [117] which satisfies

$$Q^{\mathcal{P}}_{\text{tot}}(\mathbf{h},\mathbf{a}) = \inf_{P\in\mathcal{P}} Q^{P}_{\text{tot}}(\mathbf{h},\mathbf{a}), \qquad \forall(\mathbf{h},\mathbf{a}) \in \mathcal{H}\times\mathcal{A}. \tag{8.3}$$

We call $Q^{\mathcal{P}}_{\text{tot}}$ the *optimal robust joint action-value function* for the Dec-POMDP, and it admits a deterministic robust greedy joint policy $\pi^*(\mathbf{h}) \in \arg\max_{\mathbf{a}} Q^{\mathcal{P}}_{\text{tot}}(\mathbf{h},\mathbf{a})$.

Robust Cooperative MARL. We study robust cooperative MARL under the CTDE setting. Given a model uncertainty set $\mathcal{P}$, our goal is to learn decentralized policies that maximize $Q^{\mathcal{P}}_{\text{tot}}$. That is, we aim to find $[\pi^\star_i : \mathcal{H}_i \mapsto \mathcal{A}_i]_{i\in[N]}$, such that

$$\langle \pi^\star_1, \ldots, \pi^\star_N \rangle \in \arg\max_{\mathbf{a}} Q^{\mathcal{P}}_{\text{tot}}(\cdot,\mathbf{a}).$$

Specifically, we seek a value factorization method that automatically generates robust individual action values, thereby enabling a decentralized policy. This is non-trivial for two reasons. First, no individual reward signals are available, so robust individual action values are ill-defined a priori. Second, directly defining robust individual action values from the single-agent DR-RL literature can break standard value factorization. As we demonstrate in example 3, robust individual actions may not align with the robust joint action. These challenges motivate the central question of our work: *Can we construct robust individual utilities and a mixing scheme such that decentralized greedy actions recover the joint maximizers of $Q^{\mathcal{P}}_{\text{tot}}$, thereby enabling a robust CTDE framework?*

8.3 Distributionally Robust IGM (DrIGM)

To address the question above, we propose a novel principle for robust value factorization that builds upon the IGM principle while explicitly incorporating robustness.

Distributionally Robust IGM (DrIGM) Principle

Definition 8 (DrIGM)**.** *Given an uncertainty set $\mathcal{P}$, we say that **robust** individual action-value functions $[Q_i^{\text{rob}} : \mathcal{H}_i \times \mathcal{A}_i \to \mathbb{R}]_{i\in[N]}$ satisfy the* **D**istributionally **r**obust **IGM** (DrIGM) principle *for the optimal **robust** joint action-value function $Q_{\text{tot}}^{\mathcal{P}} : \mathcal{H} \times \mathcal{A} \to \mathbb{R}$ under joint history $\mathbf{h} = (h_1, \ldots, h_N) \in \mathcal{H}$ if*

$$\left(\arg\max_{a_1} Q_1^{\text{rob}}(h_1, a_1), \ldots, \arg\max_{a_N} Q_N^{\text{rob}}(h_N, a_N)\right) \subseteq \arg\max_{\mathbf{a}} Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h}, \mathbf{a}).$$

DrIGM extends classical IGM to the robust setting by requiring that the *robust* joint greedy action induced by $Q_{\text{tot}}^{\mathcal{P}}$ factorizes into robust individual greedy actions from $[Q_i^{\text{rob}}]_{i\in[N]}$. Note that when $\mathcal{P} = \{P\}$ is a singleton (i.e., there is no uncertainty), then DrIGM is equivalent to IGM.

Satisfying DrIGM is nontrivial. In particular, the single-agent definition $Q_i^{\text{rob}}(s, a) = \inf_{P\in\mathcal{P}} Q_i^P(s, a)$, commonly adopted in the DR-RL literature, does not ensure DrIGM when applied with a global uncertainty set. As shown in Example 3 in section 8.8, an adversarial model $P \in \mathcal{P}$ that minimizes one agent's value need not coincide with the adversarial model $P' \in \mathcal{P}$ that minimizes the joint value. As a result, robust individual greedy actions may fail to align with the robust joint greedy action. Similar inconsistencies arise even under agent-wise uncertainty sets defined in Shi et al. [242] for essentially the same reason. This highlights the need for a new formulation of robust individual action values to support a consistent robust CTDE framework.

In robust cooperative MARL, the primary concern is the robustness of the *entire system*, as opposed to robustness of individual agents. Thus, it is sufficient to consider the worst case for the joint action value, rather than independently for each agent. Motivated by this idea, we show that the robust individual action value defined under a global worst-case model can guarantee DrIGM.

Theorem 10. *Given a global uncertainty set $\mathcal{P}$ defined in eq. (8.1), suppose for all $P \in \mathcal{P}$, there exist $[Q_i^P]_{i\in[N]}$ satisfying IGM for Q_{tot}^P under joint history*

$\mathbf{h} = (h_1, \ldots, h_N) \in \mathcal{H}$. *Let*

$$P^{\text{worst}}(\mathbf{h}, \mathbf{a}) \in \arg\inf_{P \in \mathcal{P}} Q_{\text{tot}}^{P}(\mathbf{h}, \mathbf{a}), \tag{8.4}$$

$$\bar{\mathbf{a}} \in \arg\max_{\mathbf{a}} Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h}, \mathbf{a}), \tag{8.5}$$

denote the global worst-case model and the robust joint greedy action, respectively. For each agent $i \in [N]$, define the robust individual action-value functions Q_i^{rob} as

$$Q_i^{\text{rob}}(h_i, a_i) := Q_i^{P^{\text{worst}}(\mathbf{h}, \bar{\mathbf{a}})}(h_i, a_i). \tag{8.6}$$

Then, $[Q_i^{\text{rob}}]_{i \in [N]}$ satisfy DrIGM for $Q_{\text{tot}}^{\mathcal{P}}$ under joint history $\mathbf{h}$.

The proof of Theorem 10 can be found in Section 8.9. Theorem 10 demonstrates that by anchoring individual robust action values to the global worst-case model evaluated at the robust joint greedy action, individual robust greedy actions become aligned with the robust joint greedy action. This construction resolves the misalignment problem that occurs when individual adversaries differ from the global adversary. More broadly, the result highlights that robust CTDE is achieved not by independently robustifying each agent, but by coordinating all agents against a shared adversarial model tied to the team's worst-case joint outcome. This perspective offers a principled foundation for designing robust value factorization methods that maintain decentralized execution while ensuring robustness guarantees.

Common factorization methods satisfy DrIGM. Having established that robust individual action values can be consistently defined via Theorem 10, we now examine whether these values are compatible with standard factorization methods used in cooperative MARL.

Theorem 11. *Given $\mathcal{P}$ defined in eq. (8.1), for a joint history $\mathbf{h} \in \mathcal{H}$, suppose for all $P \in \mathcal{P}$, there exist individual action-value functions $[Q_i^P]_{i \in [N]}$ satisfying one of the following conditions for all $\mathbf{a} = (a_1, \ldots, a_N) \in \mathcal{A}$:*

$$Q_{\text{tot}}^{P}(\mathbf{h}, \mathbf{a}) = \sum_{i \in [N]} Q_i^P(h_i, a_i), \tag{VDN}$$

$$\frac{\partial Q_{\text{tot}}^{P}(\mathbf{h}, \mathbf{a})}{\partial Q_i^P(h_i, a_i)} \geq 0, \quad \forall i \in [N], \tag{QMIX}$$

$$\sum_{i \in [N]} Q_i^P(h_i, a_i) - Q_{\text{tot}}^{P}(\mathbf{h}, \mathbf{a}) + V_{\text{tot}}(\mathbf{h}) = \begin{cases} 0, & \mathbf{a} = \bar{\mathbf{a}}, \\ \geq 0, & \mathbf{a} \neq \bar{\mathbf{a}}, \end{cases} \tag{QTRAN}$$

where $\bar{\mathbf{a}} := [\bar{a}_i]_{i\in[N]}$ *with* $\bar{a}_i := \arg\max_{a_i} Q_i^P(h_i, a_i)$ *and* $V_{\text{tot}}(\mathbf{h}) := \max_{\mathbf{a}} Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}) - \sum_{i\in[N]} Q_i^P(h_i, a_i)$ *with* $\mathbf{a} = [a_i]_{i\in[N]}$. *Then* $[Q_i^{\text{rob}}]_{i\in[N]}$ *as defined in eq.* (8.6) *satisfy DrIGM for* $Q_{\text{tot}}^{\mathcal{P}}$ *under joint history* $\mathbf{h}$.

The proof of Theorem 11 can be found in Section 8.9. Theorem 11 shows that when the underlying individual Q-functions satisfy the structural conditions of VDN [259], QMIX [217], or QTRAN [251], the robust individual action values $[Q_i^{\text{rob}}]_{i\in[N]}$ constructed from eq. (8.6) automatically satisfy DrIGM. This result ensures that robust CTDE can be realized directly within widely used value factorization frameworks, enabling principled distributionally robust extensions of existing algorithms. Moreover, as long as the test environment lies within the prescribed uncertainty set, this approach yields a provable robustness guarantee, as formalized in the next theorem.

Theorem 12. *Given* $\mathcal{P}$ *defined in eq.* (8.1)*, suppose the robust individual action values* Q_i^{rob} *satisfy definition 8. If the test environment model* P_{test} *is included in the uncertainty set (i.e.,* $P_{\text{test}} \in \mathcal{P}$*), then the robust joint action values provably lower bound the real joint action values in* P_{test}*:*

$$Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h}, \mathbf{a}) \le Q_{\text{tot}}^{P_{\text{test}}}(\mathbf{h}, \mathbf{a}), \quad \forall \mathbf{h} \in \mathcal{H},\ \mathbf{a} \in \mathcal{A}.$$

The proof of Theorem 12 can be found in Section 8.9.

Robust Bellman Operators under Specific Uncertainty Sets

To design training loss functions, we next present the DrIGM-based robust Bellman operators for two common uncertainty designs: ρ-contamination and total variation (TV), which are well-studied in the single-agent distributionally robust RL literature [303, 197, 301, 72, 167, 198, 288, 321]. Both types of uncertainty sets consider perturbations of size $\rho \in (0, 1]$ around a nominal model P^0.

The ρ-contamination uncertainty set is defined as (for all $\mathbf{h} \in \mathcal{H}$ and $\mathbf{a} \in \mathcal{A}$)

$$\mathcal{P}_{\mathbf{h},\mathbf{a}} = \left\{ P \in \Delta(\mathcal{H}) \,\middle|\, P_{\mathbf{h},\mathbf{a}} = (1-\rho) P^0_{\mathbf{h},\mathbf{a}} + \rho \nu_{\mathbf{h},\mathbf{a}},\ \nu \in \Delta(\mathcal{H}) \text{ is arbitrary} \right\}, \quad (8.7)$$

with corresponding robust Bellman operator

$$\begin{aligned}(\mathcal{T} Q_{\text{tot}}^{\mathcal{P}})(\mathbf{h}, \mathbf{a}) &\overset{(a)}{=} r(s, \mathbf{a}) + \gamma(1-\rho)\, \mathbb{E}_{\mathbf{h}' \sim P^0_{\mathbf{h},\mathbf{a}}} \left[\max_{\mathbf{a}' \in \mathcal{A}} Q_{\text{tot}}^{\mathcal{P}}(h'_1, \ldots, h'_N, \mathbf{a}') \right] \\ &\overset{(b)}{=} r(s, \mathbf{a}) + \gamma(1-\rho)\, \mathbb{E}_{\mathbf{h}' \sim P^0_{\mathbf{h},\mathbf{a}}} \left[Q_{\text{tot}}^{\mathcal{P}}(h'_1, \ldots, h'_N, \bar{a}'_1, \ldots, \bar{a}'_N) \right], \end{aligned} \quad (8.8)$$

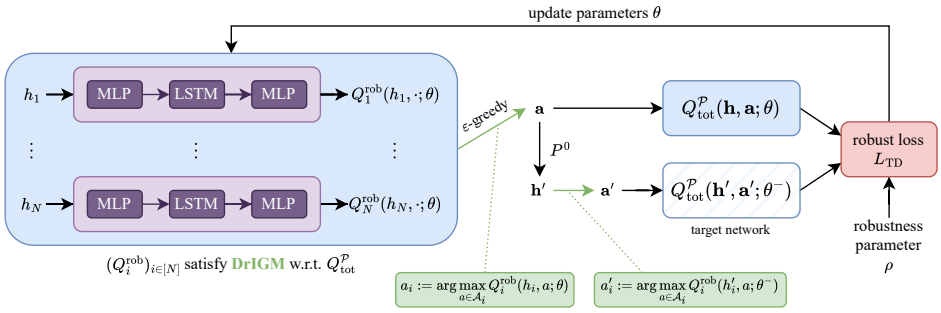


Figure 8.1: Overview of our robust value factorization algorithms. Because the robust individual action-value functions satisfy DrIGM, greedy actions can be computed efficiently in a decentralized manner while the function parameters are trained with a robust TD loss based on global reward.

where $\bar{a}_i' = \arg\max_{a_i'} Q_i^{\text{rob}}(h_i', a_i')$. Here, (a) follows from robust Bellman operator as in the single-agent setting due to the $\mathbf{h} \times \mathbf{a}$-rectangularity from eq. (8.1). (b) follows from the DrIGM principle where robust individual greedy actions are aligned with the robust joint greedy action.

Similarly, the TV-uncertainty set is defined as (for all $\mathbf{h} \in \mathcal{H}$ and $\mathbf{a} \in \mathcal{A}$)

$$\mathcal{P}_{\mathbf{h},\mathbf{a}} = \left\{ P \in \Delta(\mathcal{H}) \,\middle|\, \text{TV}(P, P^0_{\mathbf{h},\mathbf{a}}) \leq \rho \right\}, \tag{8.9}$$

with corresponding robust Bellman operator,

$$\begin{aligned}(\mathcal{T}Q_{\text{tot}}^{\mathcal{P}})(\mathbf{h}, \mathbf{a}) = r(s, \mathbf{a}) - \inf_{\eta \in [0, \frac{2}{\rho(1-\gamma)}]} \gamma\, \mathbb{E}_{\mathbf{h}' \sim P^0_{\mathbf{h},\mathbf{a}}} \Bigg(&- (1-\rho)\eta(s, \mathbf{a}) \\ &+ \left[\eta(s, \mathbf{a}) - Q_{\text{tot}}^{\mathcal{P}}(h_1', \ldots, h_N', \bar{a}_1', \ldots, \bar{a}_N') \right]_+ \Bigg),\end{aligned} \tag{8.10}$$

where $\eta(s, a)$ is the dual variable.

The derivation of the robust Bellman operators relies on a "fail state" assumption (assumption 12); the proof is provided in section 8.10. In the next section, we show how DrIGM leads to practical robust value factorization algorithms.

8.4 Algorithms: Robust Value Factorization

Overall framework. Guided by DrIGM, we develop six robust value factorization algorithms by combining two types of uncertainty sets (ρ-contamination, TV-uncertainty) with three different value factorization architectures (VDN, QMIX, QTRAN). The overall framework is illustrated in algorithm 7 and fig. 8.1, while detailed pseudocode for each variant can be found in section 8.11. Concretely,

we collect trajectories using ε-greedy exploration and train the robust individual action-value network using TD-learning [260]. For stability, robust one-step targets are evaluated using *target* networks, which are updated periodically.

Algorithm 7 Robust value factorization

Input robustness parameter ρ, target network update frequency f, and ε
Initialize replay buffer $\mathcal{D}$
Initialize **robust individual action-value networks** $[Q_i^{\text{rob}}]_{i\in[N]}$ with random parameters θ
Initialize **factorization networks** that produce $Q_{\text{tot}}^{\mathcal{P}}$ with random parameters θ
Initialize target parameters $\theta^- = \theta$
for episode $e = 1, \ldots, E$ **do**
    Observe initial state s^0 and observation $o_i^0 = \sigma_i(s^0)$ for each agent i.
    for $t = 1, \ldots, T$ **do**
        Each agent i chooses its action a_i^t using ε-greedy policy.
        Take joint action $\mathbf{a}^t$, observe the next state s^{t+1}, reward r^t and observation $o_i^{t+1} = \sigma_i(s^{t+1})$ for each agent i
        Store transition $(\mathbf{h}^t, \mathbf{a}^t, r^t, \mathbf{h}^{t+1})$ in replay buffer $\mathcal{D}$
        Sample a mini-batch of transitions $(\mathbf{h}, \mathbf{a}, r, \mathbf{h}')$ from $\mathcal{D}$
        Calculate **TD loss** L_{TD} using eq. (8.14)
        Update θ by minimizing L_{TD}
        Update $\theta^- = \theta$ with frequency f

Robust individual action-value networks. Each agent i uses a DRQN (Deep Recurrent Q Network)-style network that maps its local history h_i (observations and past actions) to action-values $Q_i(h_i, a_i)$, following an *MLP* encoder $\rightarrow$ *LSTM* core $\rightarrow$ *MLP* output architecture. For our training procedure, we follow the approach from Hausknecht and Stone [105]. We sample mini-batches of *sub-trajectories* from the replay buffer $\mathcal{D}$ and use *bootstrapped random updates*. We use 8 burn-in steps to warm-start the LSTM state and only take the last step output to calculate the loss and update the networks. This procedure is computationally and memory efficient while achieving performance comparable to sequential updates from the start of each episode.

Factorization networks. We instantiate three networks for robust value factorization:

1. **VDN** factorizes the robust joint action-value as the sum of robust per-agent

values,

$$Q_{\text{tot}}^{\mathcal{P},\text{VDN}}(\mathbf{h},\mathbf{a}) \;=\; \sum_{i=1}^{N} Q_i^{\text{rob}}(h_i,a_i).$$

2. Beyond direct summation, **QMIX** uses a *monotone* mixing network

$$Q_{\text{tot}}^{\mathcal{P},\text{QMIX}}(\mathbf{h},\mathbf{a}) \;=\; f_\theta\big(Q_1^{\text{rob}}(h_1,a_1),\ldots,Q_N^{\text{rob}}(h_N,a_N),\, s\big), \tag{8.11}$$

where s is the global state. A lightweight *hypernetwork* takes s as input and outputs the layer weights of f_θ; to ensure $\partial Q_{\text{tot}}^{\mathcal{P},\text{QMIX}}/\partial Q_i^{\text{rob}} \geq 0$ (the QMIX monotonicity constraint), we enforce elementwise nonnegativity on these weights via an absolute-value (or softplus) transform. Biases remain unconstrained.

3. **QTRAN** learns a separate joint action-value function $Q_{\text{tot}}^{\mathcal{P},\text{QTRAN}}(\mathbf{h},\mathbf{a})$ and a baseline $V_{\text{tot}}(\mathbf{h})$. For efficiency and scalability, the joint network shares the encoder/head with the individual DRQN modules. In addition to the robust TD loss, QTRAN imposes two *consistency* terms to align the factorized and joint values:

$$L_{\text{opt}} \;=\; \Big(Q_{\text{tot}}^{\mathcal{P},\text{VDN}}(\mathbf{h},\bar{\mathbf{a}}) - \widehat{Q}_{\text{tot}}^{\mathcal{P},\text{QTRAN}}(\mathbf{h},\bar{\mathbf{a}}) + V_{\text{tot}}(\mathbf{h})\Big)^2, \tag{8.12}$$

$$L_{\text{nopt}} \;=\; \Big(\min\big[Q_{\text{tot}}^{\mathcal{P},\text{VDN}}(\mathbf{h},\mathbf{a}) - \widehat{Q}_{\text{tot}}^{\mathcal{P},\text{QTRAN}}(\mathbf{h},\mathbf{a}) + V_{\text{tot}}(\mathbf{h}),\, 0\big]\Big)^2, \tag{8.13}$$

where the $\hat{Q}$ is the detached Q value for training stability. Intuitively, L_{opt} enforces equality at the (robust) greedy joint action, while L_{nopt} penalizes positive slack elsewhere, recovering the QTRAN constraints in our robust setting.

TD Loss. Given the robust Bellman operator $\mathcal{T}$ defined in eq. (8.2) for a Dec-POMDP setting, the generic form of the TD-loss is

$$L_{\text{TD}} = \Big(Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h},\mathbf{a};\theta) - (\mathcal{T}Q_{\text{tot}}^{\mathcal{P}}(\cdot,\cdot;\theta^-))(\mathbf{h},\mathbf{a})\Big)^2, \tag{8.14}$$

where θ is the network parameters, and θ^- is the target network parameters for training stability. Specifically, for ρ-contamination uncertainty sets, by the robust Bellman operator in eq. (8.8), we have

$$L_{\text{TD}} = \Big(Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h},\mathbf{a};\theta) - (r(s,\mathbf{a}) + \gamma(1-\rho)\,\mathbb{E}_{\mathbf{h}'\sim P^0_{\mathbf{h},\mathbf{a}}}\, Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h}',\bar{\mathbf{a}}';\theta^-))\Big)^2, \tag{8.15}$$

For TV uncertainty sets, by the robust Bellman operator in eq. (8.10), we have

$$L_{\text{TD}} = \Bigg(Q^{\mathcal{P}}_{\text{tot}}(\mathbf{h}, \mathbf{a}; \theta) - r(s, \mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{h}' \sim P^0_{\mathbf{h},\mathbf{a}}} \Bigg[-(1-\rho)\eta(s, \mathbf{a}) + [\eta(s, \mathbf{a}) - Q^{\mathcal{P}}_{\text{tot}}(\mathbf{h}', \bar{\mathbf{a}}'; \theta^-)]_+ \Bigg] \Bigg)^2, \tag{8.16}$$

where $\eta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is calculated by minimizing the following empirical loss:

$$L_{\text{dual}}(\eta, Q^{\mathcal{P}}_{\text{tot}}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{h},\mathbf{a},\mathbf{h}') \in \mathcal{D}} \left(\left[\eta(s, \mathbf{a}) - \max_{\mathbf{a}'} Q^{\mathcal{P}}_{\text{tot}}(\mathbf{h}', \mathbf{a}') \right]_+ - (1-\rho)\, \eta(s, \mathbf{a}) \right). \tag{8.17}$$

8.5 Experiments

SustainGym

We evaluate our proposed robust value factorization methods on SustainGym [310], a recent benchmark suite designed to simulate real-world control tasks under distribution shift. We focus on multi-agent environments for smart building HVAC control, which inherently involve stochastic dynamics, distribution shifts, partial observability, and inter-agent coupling. These environments are particularly well-suited to test robustness, as the environmental models can vary across days and building locations (thus climate conditions). More details about the environment and the experiment setup are provided in section 8.12. Our code can be found at https://github.com/crqu/robust-coMARL.

Evaluation protocol. To assess generalization under distribution shift (i.e., model uncertainty), we adopt the following protocol. In the **training phase**, each algorithm is trained on a single environment. For robust MARL baselines that require multiple environments, we follow their standard protocol and train them on a fixed set of environments. In the **evaluation phase**, trained policies are deployed on unseen configurations that differ from those used in training, simulating realistic deployment where distribution shifts inevitably arise. This design allows us to explicitly measure robustness to changes in environment dynamics rather than simple memorization of training conditions.

Baselines. We compare our robust value factorization methods against:

- Non-robust factorization methods: VDN, QMIX, and QTRAN trained without robustness considerations, representing the standard CTDE paradigm.

- Existing robust CTDE baseline: the multi-agent group distributionally robust algorithm from Liu et al. [165], which we refer to as "GroupDR". While the original work used only the VDN architecture, we extend the algorithm to QMIX and QTRAN for completeness.

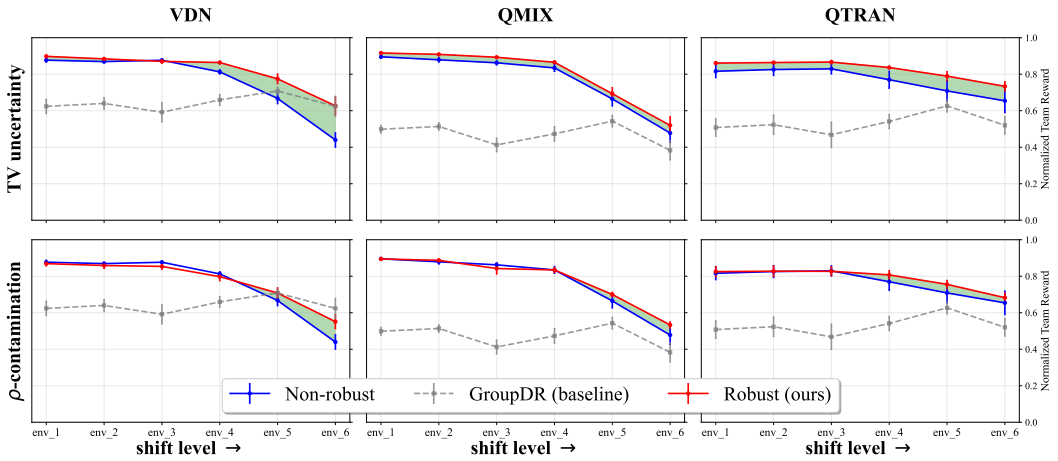


Figure 8.2: Normalized performance (averaged over 5 independent training runs, with error bars showing standard error) across different environment configurations for our robust MARL algorithms and other baselines. Each panel corresponds to one value factorization method. Robustness gain is the difference in reward (shaded area) between Robust (ours) and Non-robust, which shows the out-of-distribution performance improvement from the robust training.

Experiment 1: climatic shifts. We first test robustness under shifts induced by changes in climate conditions. Results (averaged over five seeds) are shown in fig. 8.2. Our robust MARL algorithms consistently outperform both non-robust counterparts and the group DR baseline. Notably, performance degradation scales with the severity of the shift (e.g., `env_6` deviates most from the training environment, `env_1`), but our methods maintain relatively high returns. In contrast, the GroupDR baseline exhibits little sensitivity to shift severity, reflecting its reliance on worst-case rewards computed only from configurations encountered during training.

Experiment 2: seasonal shifts. We next evaluate robustness to seasonal shifts, training algorithms on `season_1` data and evaluating on `season_2`. Results are reported in table 8.1, showing mean and standard error of normalized episodic returns. The results show that robust value factorization algorithms with TV uncertainty set achieve consistent robustness gain against seasonal shifts.

Table 8.1: Performance under seasonal shifts for our robust MARL algorithms and other baselines (mean ± standard error over 5 independent training runs). Values outperforming both the non-robust and group DR baselines are highlighted in bold.

Factorization Methods	VDN	QMIX	QTRAN
Non-robust	0.877 ± 0.012	0.895 ± 0.008	0.816 ± 0.036
baseline (GroupDR)	0.624 ± 0.040	0.499 ± 0.022	0.508 ± 0.048
Robust (TV-uncertainty)	**0.898 ± 0.008**	**0.916 ± 0.006**	**0.861 ± 0.006**
Robust (ρ-contamination)	0.869 ± 0.013	**0.911 ± 0.005**	**0.825 ± 0.028**

Table 8.2: Performance under climatic and seasonal shifts for our robust MARL algorithms and other baselines (mean ± standard error over 5 independent training runs). Values outperforming both the non-robust and group DR baselines are highlighted in bold.

Factorization Methods	VDN	QMIX	QTRAN
Non-robust	0.440 ± 0.040	0.478 ± 0.052	0.654 ± 0.066
baseline (GroupDR)	0.624 ± 0.056	0.383 ± 0.053	0.520 ± 0.049
Robust (TV-uncertainty)	**0.627** ± 0.049	**0.520 ± 0.048**	**0.733 ± 0.026**
Robust (ρ-contamination)	0.551 ± 0.039	**0.500** ± 0.075	**0.682 ± 0.026**

Experiment 3: climatic and seasonal shifts. Finally, we test on the most extreme case, where we have distribution shifts arising from climatic and seasonal shifts. The results are presented in table 8.2, with our robust MARL algorithms achieving 10-40% higher average reward than the non-robust baseline. Notably, QTRAN-based robust MARL algorithms demonstrate strong out-of-distribution performance and stability.

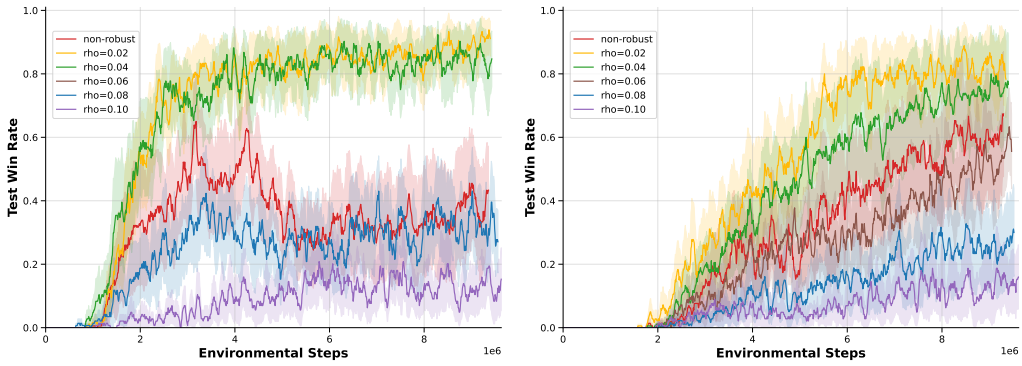
Choice of ρ. Theoretically, ρ should be chosen based on prior estimation of the model uncertainty level. Practically, we select ρ by training on `env_1` and validating on `env_2` and `env_3`, which yields stable performance without overfitting to a single shift.

Robustness in cooperative MARL. A noteworthy finding is that robustness in cooperative MARL does *not necessarily* entail reduced performance in the training environment. Unlike in single-agent robust RL, where conservatism often penalizes in-distribution returns, explicitly modeling robustness here mitigates errors from partial observability and decentralized execution. In several cases, robust training even improves in-distribution performance relative to non-robust baselines, suggesting

that robustness can simultaneously enhance stability and adaptability in multi-agent systems.

StarCraft II

We additionally conduct experiments in SMAC [228], a well-known benchmark consisting of two teams of agents engaged in cooperative combat scenarios based on StarCraft II. We focus on the hard `3s_vs_5z` map. In the test environment, we introduce distribution shift by adding noise to each agent's observation of every enemy unit's normalized position, sampled from $\mathcal{N}(0, 0.75^2)$. Since QTRAN has been shown to perform poorly on SMAC [250], we report results for the ρ-contamination uncertainty set using only VDN and QMIX (averaged over five seeds) in fig. 8.3. The results demonstrate that for small values of ρ, our robust MARL algorithms significantly improve out-of-distribution performance.

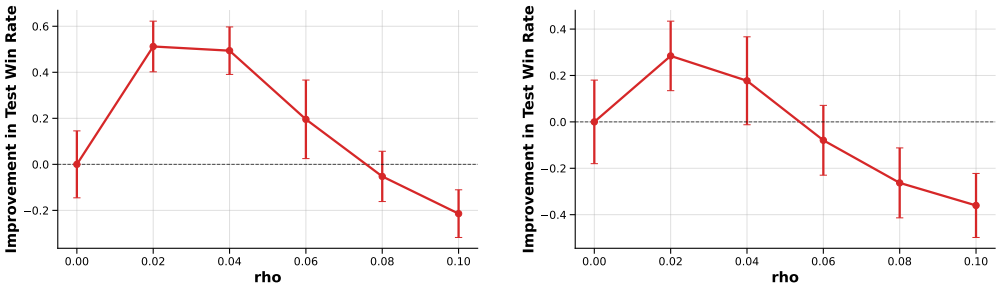


(a) Robust VDN (b) Robust QMIX

Figure 8.3: Performance of our robust MARL algorithms and their non-robust baselines in SMAC (`3s_vs_5z` map). Each algorithm is evaluated every 10,000 environment steps, with each evaluation averaged over 32 episodes. Shaded regions denote the standard error across 5 random seeds. For small ρ, the robust algorithms significantly outperform their non-robust counterparts.

Robustness parameter evaluation. We further compare the final performance of our algorithms against their non-robust baselines, reporting the improvement in the test win rate for different choices of ρ relative to the baseline. The results (averaged over five seeds) are shown in fig. 8.4. Interestingly, the test win rate first increases as ρ grows, and then decreases. This observation aligns with our theory: when ρ is small relative to the shift level, explicitly modeling distribution shift during training yields improved out-of-distribution performance. However, when ρ becomes

large, the robust MARL algorithms become overly conservative, leading to degraded performance.



(a) Robust VDN (b) Robust QMIX

Figure 8.4: Improvement in final test win rate of our robust MARL algorithms over their non-robust baselines in SMAC (`3s_vs_5z` map) for different values of ρ. Error bars denote the standard error across 5 random seeds.

8.6 Conclusion

In this work, we introduce Distributionally robust IGM (DrIGM), a robustness principle for cooperative MARL that extends the classical IGM property to settings with environmental uncertainties. Whereas naïvely "robustifying" individual agent policies fails to align robust individual policies with the joint robust policy, the DrIGM offers a principled framework for constructing robust individual action values that remain aligned with the joint robust policy, thereby enabling decentralized greedy execution under uncertainty.

Building on this foundation, we derive DrIGM-based robust value factorization algorithms for VDN, QMIX, and QTRAN, trained via robust Bellman operators under standard uncertainty sets (ρ-contamination and total variation). Empirically, on a high-fidelity building HVAC control benchmark, our methods consistently mitigate out-of-distribution performance degradation arising from climatic and seasonal shifts. On a StarCraft II game-playing benchmark, our methods likewise improve out-of-distribution performance under added observation noise. Unlike single-agent robust RL, where conservatism often harms in-distribution returns, we find that robustness in cooperative MARL can simultaneously enhance stability and adaptability.

While we introduced the DrIGM framework for a global uncertainty set, we believe it may be possible to further extend this framework. Future work includes developing DrIGM-compliant algorithms under agent-wise uncertainty sets and

exploring additional training paradigms (e.g., decentralized training) to further broaden applicability.

Reproducibility statement

We release a GitHub repository containing all code, configuration files, and scripts needed to reproduce our results, including data generation and figure plotting. All proofs for the main paper are stated in Section 8.9. Algorithm pseudocode is also provided in Section 8.11.

8.7 Related Work

Single-agent Distributionally Robust RL (DR-RL). The single-agent setting is typically formalized as a robust Markov decision process (MDP). A substantial literature studies finite-sample guarantees for distributionally robust RL, exploring a variety of ambiguity-set designs [117, 299, 294, 128, 109, 247, 110, 98, 68, 262, 196, 224, 67, 176]. Most relevant to our work are tabular robust MDPs with (s, a)-rectangular uncertainty sets defined by total-variation balls [303, 197, 301, 72, 167, 198] or ρ-contamination models [288, 321], for which minimax dynamic programming and learning algorithms admit provable performance bounds.

Value factorization methods for cooperative MARL. Value factorization is the standard mechanism for scalable cooperative MARL under CTDE. Early work adopts simple additivity (VDN [259]), while QMIX [217] learn a state-conditioned monotone combiner to enlarge the function class without violating the IGM requirement. QTRAN [251] further relax the monotonicity assumption with consistency constraints. Other approaches include attention-based mixers (e.g., QAtten [304], REFIL [116]), dueling-style decompositions (QPlex [282]) and residual designs (ResQ [240]). Building on this body of work, we develop robust value-factorization algorithms with provable robustness guarantees under model uncertainty, enabling robust decentralized execution in partially observable cooperative settings.

Robustness in MARL. In general MARL, robustness is typically studied within Markov games, where uncertainty can be modeled in different components, such as the state space [104, 108, 327, 322], other agents [153, 125], and environmental dynamics [319, 165]. We refer readers to Vial et al. [275] for an overview. This work considers robustness to model uncertainty, primarily studied via distributionally robust optimization (DRO) [216, 92, 29, 75, 36, 87, 208, 179, 192], where most prior efforts target Nash equilibria and provide provable (actor-critic / Q-learning)

algorithms [319, 127, 170, 35, 242, 165], often under full observability or individually rewarded settings. We complement this line by addressing the cooperative, partially observable CTDE regime, where agents receive a single joint reward and act only local observations.

In cooperative MARL, robustness has been modeled along several complementary axes, including adversarial (Byzantine) teammates [157], state/observation disturbances [100], communication errors [317], risk-sensitive objectives that guard against tail events under a fixed model [239], and explicit model uncertainty [143, 320]. Focusing on the last category, Kwak et al. [143] address model uncertainty with sparse, execution-time communication, whereas Zhang et al. [320] study settings in which each agent observes the full state and receives individual reward. Similarly, Bukharin et al. [47] also considers settings where each agent receives individual reward, and they achieve robustness by controlling the Lipschitz constant of each agent's policy. In contrast, our work targets robustness to model uncertainty in the cooperative CTDE setting, complementing prior approaches by providing a systematic framework that does *not* require real-time communication and operates under partial observability with a single team reward.

8.8 Examples

Example 3 (Naïve single-agent robust action values cannot guarantee DrIGM)**.** *Consider a robust cooperative two-agent task (illustrated in fig. 8.5) with action spaces* $\mathcal{A}_1 = \mathcal{A}_2 = \{1, 2\}$*, state space* $\mathcal{S} = \{s_0, s_1, s_2, s_3, s_4\}$*, and uncertainty set* $\mathcal{P} = \{P_1, P_2\}$*. Let* s_0 *be the initial state, and let* s_1*,*s_2*,*s_3 *and* s_4 *all be absorbing states with zero reward. We assume each agent observes the full state. For* P_1*, all the transitions are fully deterministic.* P_2 *differs from* P_1 *only in transitions on joint actions* $(1, 2)$ *and* $(2, 1)$*, given by:*

$$\mathbb{P}(S_2 \mid 1, 2) = \frac{1}{3}, \mathbb{P}(S_3 \mid 1, 2) = \frac{2}{3},$$
$$\mathbb{P}(S_2 \mid 2, 1) = \frac{2}{3}, \mathbb{P}(S_3 \mid 2, 1) = \frac{1}{3},$$

As shown in fig. 8.5, the optimal joint action value function at state s_0 *is (we omit the* $\gamma/(1-\gamma)$ *factor for clarity)*

$$Q_{\text{tot}}^{P_1}(s_0, 1, 1) = 0.7, \quad Q_{\text{tot}}^{P_1}(s_0, 1, 2) = 0.4, \quad Q_{\text{tot}}^{P_1}(s_0, 2, 1) = 1.0, \quad Q_{\text{tot}}^{P_1}(s_0, 2, 2) = 0.7;$$
$$Q_{\text{tot}}^{P_2}(s_0, 1, 1) = 0.7, \quad Q_{\text{tot}}^{P_2}(s_0, 1, 2) = 0.8, \quad Q_{\text{tot}}^{P_2}(s_0, 2, 1) = 0.6, \quad Q_{\text{tot}}^{P_2}(s_0, 2, 2) = 0.7.$$

Therefore, the robust joint action-value function $Q^{\mathcal{P}}_{\text{tot}}(s, \mathbf{a}) = \inf_{P \in \mathcal{P}} Q^{\mathcal{P}}_{\text{tot}}(s, \mathbf{a})$ *is given by:*

$$Q^{\mathcal{P}}_{\text{tot}}(s_0, 1, 1) = 0.7, \quad Q^{\mathcal{P}}_{\text{tot}}(s_0, 1, 2) = 0.4, \quad Q^{\mathcal{P}}_{\text{tot}}(s_0, 2, 1) = 0.6, \quad Q^{\mathcal{P}}_{\text{tot}}(s_0, 2, 2) = 0.7,$$

It is straightforward to check that the following individual action-value functions $\{Q_i^{P_j}\}_{i,j \in [2]}$ *satisfy* $Q^P_{\text{tot}}(s, a_1, a_2) = Q^P_1(s, a_1) + Q^P_2(s, a_2)$ *for all* $s \in \mathcal{S}$ *and* $P \in \mathcal{P}$*, which is a special case of the classical IGM property:*

$$\begin{aligned} &Q_1^{P_1}(s_0, 1) = 0, \quad &&Q_1^{P_1}(s_0, 2) = 0.3, \quad &&Q_2^{P_1}(s_0, 1) = 0.7, \quad &&Q_2^{P_1}(s_0, 2) = 0.4, \\ &Q_1^{P_2}(s_0, 1) = 0.2, \quad &&Q_1^{P_2}(s_0, 2) = 0.1, \quad &&Q_2^{P_2}(s_0, 2) = 0.5, \quad &&Q_2^{P_2}(s_0, 2) = 0.6. \end{aligned}$$

Suppose the robust individual action-value function is defined as $Q_i^{\text{rob}}(s, a) = \inf_{P \in \mathcal{P}} Q_i^P(s, a)$*, as in the single-agent DR-RL literature. Therefore, the robust individual action-value functions are given by:*

$$Q_1^{\text{rob}}(s_0, 1) = 0, \quad Q_1^{\text{rob}}(s_0, 2) = 0.1, \quad Q_2^{\text{rob}}(s_0, 1) = 0.5, \quad Q_2^{\text{rob}}(s_0, 2) = 0.4,$$

At s_0*, these robustifications fail to satisfy DrIGM:*

$$\begin{aligned} (2, 1) &= \left(\arg\max_{a_1} Q_1^{\text{rob}}(s_0, a_1), \arg\max_{a_2} Q_2^{\text{rob}}(s_0, a_2) \right) \\ &\notin \arg\max_{\mathbf{a}} Q^{\mathcal{P}}_{\text{tot}}(s_0, \mathbf{a}) = \{(1, 1), (2, 2)\}. \end{aligned}$$

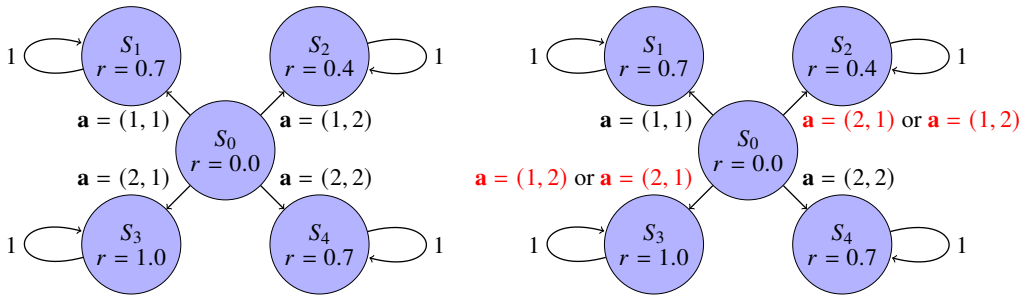


Figure 8.5: **(a)** MDP under transition kernel P_1, **(b)** MDP under transition kernel P_2. The two differ in their transition probabilities to s_2 and s_3.

Example 4 (DrIGM can address cases where IGM fails.)**.** *Consider a similar robust cooperative two-agent task (illustrated in fig. 8.6) with action spaces* $\mathcal{A}_1 = \mathcal{A}_2 = \{1, 2\}$*, state space* $\mathcal{S} = \{s_0, s_1, s_2, s_3, s_4\}$*, and uncertainty set* $\mathcal{P} = \{P_1, P_2\}$*. Let* P_1 *be the training and testing environment. For* P_1*, all the transitions are fully*

deterministic, the optimal joint action value function at state s_0 is (we omit the $\gamma/(1-\gamma)$ factor for clarity):

$$Q_{\text{tot}}^{P_1}(s_0,1,1)=0.7,\ Q_{\text{tot}}^{P_1}(s_0,1,2)=0.4,\quad Q_{\text{tot}}^{P_1}(s_0,2,1)=1.0,\ Q_{\text{tot}}^{P_1}(s_0,2,2)=0.5.$$

P_2 differs from P_1 in that all actions leads to S_4. Therefore, the optimal joint action value function at state s_0 is (we omit the $\gamma/(1-\gamma)$ factor for clarity):

$$Q_{\text{tot}}^{P_2}(s_0,1,1)=0.4,\ Q_{\text{tot}}^{P_2}(s_0,1,2)=0.4,\quad Q_{\text{tot}}^{P_2}(s_0,2,1)=0.4,\ Q_{\text{tot}}^{P_2}(s_0,2,2)=0.4.$$

It can be verified that P_1 does not admit a VDN-style value decomposition, but the worst case, P_2, admits a feasible VDN-style value decomposition.

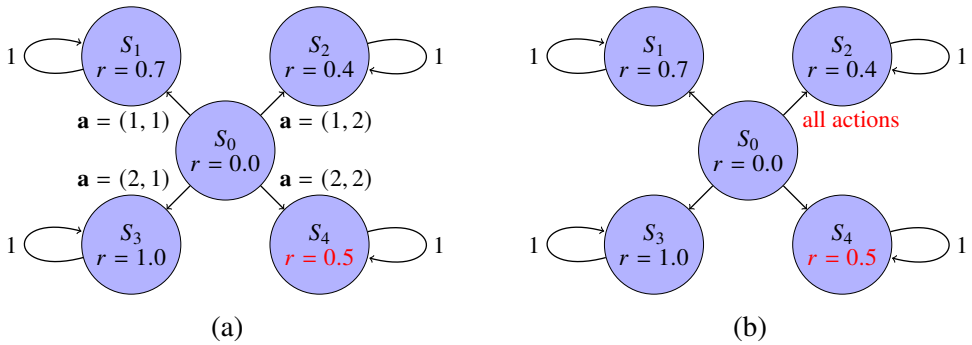


Figure 8.6: **(a)** MDP under transition kernel P_1, **(b)** MDP under transition kernel P_2.

8.9 Proofs

Proof of Theorem 10

Proof. Recall that for all $\mathbf{a} \in \mathcal{A}$, we have

$$Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h},\mathbf{a}) = \inf_{P\in\mathcal{P}} Q_{\text{tot}}^{P}(\mathbf{h},\mathbf{a}) \qquad \text{(eq. (8.3))}$$

$$P^{\text{worst}}(\mathbf{h},\mathbf{a}) \in \arg\inf_{P\in\mathcal{P}} Q_{\text{tot}}^{P}(\mathbf{h},\mathbf{a}). \qquad \text{(eq. (8.4))}$$

Thus, $Q_{\text{tot}}^{P^{\text{worst}}(\mathbf{h},\mathbf{a})}(\mathbf{h},\mathbf{a}) = Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h},\mathbf{a})$. In eq. (8.4), we also defined

$$\bar{\mathbf{a}} \in \arg\max_{\mathbf{a}\in\mathcal{A}} Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h},\mathbf{a}).$$

Since $P^{\text{worst}}(\mathbf{h}, \bar{\mathbf{a}}) \in \mathcal{P}$, by assumption there exist $[Q_i^{P^{\text{worst}}(\mathbf{h},\bar{\mathbf{a}})}]_{i\in[N]}$ that satisfy IGM for $Q_{\text{tot}}^{P^{\text{worst}}(\mathbf{h},\bar{\mathbf{a}})}$ under $\mathbf{h}$. Therefore,

$$
\begin{aligned}
&\left(\arg\max_{a_1} Q_1^{\text{rob}}(h_1, a_1), \ldots, \arg\max_{a_N} Q_N^{\text{rob}}(h_N, a_N)\right) \\
&= \left(\arg\max_{a_1} Q_1^{P^{\text{worst}}(\mathbf{h},\bar{\mathbf{a}})}(h_1, a_1), \ldots, \arg\max_{a_N} Q_N^{P^{\text{worst}}(\mathbf{h},\bar{\mathbf{a}})}(h_N, a_N)\right) && \text{(eq. (8.4))} \\
&\subseteq \arg\max_{\mathbf{a}} Q_{\text{tot}}^{P^{\text{worst}}(\mathbf{h},\bar{\mathbf{a}})}(\mathbf{h}, \mathbf{a}) && \text{(IGM)} \\
&\subseteq \arg\max_{\mathbf{a}} Q_{\text{tot}}^{P^{\text{worst}}(\mathbf{h},\mathbf{a})}(\mathbf{h}, \mathbf{a}) && \text{(eq. (8.4))} \\
&= \arg\max_{\mathbf{a}} Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h}, \mathbf{a}),
\end{aligned}
$$

which shows that $[Q_i^{\text{rob}}]_{i\in[N]}$ satisfy DrIGM under $\mathbf{h}$. □

Remark 2. *The statement of Theorem 10 assumes that for all $P \in \mathcal{P}$ there exist $[Q_i^P]_{i\in[N]}$ satisfying IGM for Q_{tot}^P under joint history $\mathbf{h} \in \mathcal{H}$. However, we note here that this assumption can be relaxed to only requiring that there exist $[Q_i]_{i\in[N]}$ satisfying IGM for $Q_{\text{tot}}^{P^{\text{worst}}}$ under $\mathbf{h}$.*

Remark 3. *As shown in Example 3, an adversarial model $P \in \mathcal{P}$ that minimizes one agent's value need not coincide with the adversarial model $P' \in \mathcal{P}$ that minimizes the joint value. Theorem 10 circumvents this problem by directly considering the global worst case, i.e.,*

$$Q_i^{\text{rob}}(h_i, a_i) := Q_i^{P^{\text{worst}}(\mathbf{h},\bar{\mathbf{a}})}(h_i, a_i). \tag{8.18}$$

Given this definition, a robust joint action is given by $\bar{\mathbf{a}} = (1, 1)$ or $(2, 2)$, and there exist robust individual action-value functions given by:

$$Q_1^{\text{rob}}(s_0, 1) = 0.2, \quad Q_1^{\text{rob}}(s_0, 2) = 0.1, \quad Q_2^{\text{rob}}(s_0, 1) = 0.7, \quad Q_2^{\text{rob}}(s_0, 2) = 0.4,$$

where the robust individual action is given by $a_1 = 1, a_2 = 1$. Alternatively, another set of individual action-value functions are given by:

$$Q_1^{\text{rob}}(s_0, 1) = 0, \quad Q_1^{\text{rob}}(s_0, 2) = 0.3, \quad Q_2^{\text{rob}}(s_0, 1) = 0.5, \quad Q_2^{\text{rob}}(s_0, 2) = 0.6,$$

where the robust individual action is given by $a_1 = 2, a_2 = 2$. Either way, the robust individual actions are aligned with the joint actions.

Proof of Theorem 11

Proof. We proceed by proving that IGM holds for under each of the three conditions given in Theorem 11. By Theorem 10, it suffices to show that DrIGM holds for each of the three conditions given in Theorem 11.

VDN condition. Given a joint history $\mathbf{h} \in \mathcal{H}$, for any $P \in \mathcal{P}$, we have

$$Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}) = \sum_{i \in [N]} Q_i^P(h_i, a_i), \quad \forall \mathbf{a} = (a_1, \ldots, a_N) \in \mathcal{A}.$$

Let $\bar{a}_i = \arg\max_{a_i} Q_i^P(h_i, a_i)$ for $i \in [N]$, and let $\bar{\mathbf{a}} = [\bar{a}_i]_{i \in [N]}$. Then, for any $\mathbf{a} \in \mathcal{A}$,

$$\begin{aligned} Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}) &= \sum_{i \in [N]} Q_i^P(h_i, a_i), \\ &\leq \sum_{i \in [N]} Q_i^P(h_i, \bar{a}_i) && \text{(Definition of } \bar{a}_i\text{)} \\ &= Q_{\text{tot}}^P(\mathbf{h}, \bar{\mathbf{a}}). \end{aligned}$$

This implies that

$$\left(\arg\max_{a_1} Q_1^P(h_1, a_1), \ldots, \arg\max_{a_N} Q_N^P(h_N, a_N) \right) \subseteq \arg\max_{\mathbf{a}} Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}),$$

so $[Q_i^P]_{i \in [N]}$ satisfy IGM for Q_{tot}^P under $\mathbf{h}$. Therefore, by Theorem 10, $[Q_i^{\text{rob}}]_{i \in [N]}$ satisfy DrIGM for $Q_{\text{tot}}^{\mathcal{P}}$ under $\mathbf{h}$.

QMIX condition. Given a joint history $\mathbf{h} \in \mathcal{H}$, for any $P \in \mathcal{P}$, suppose the following monotonicity property holds:

$$\frac{\partial Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a})}{\partial Q_i^P(h_i, a_i)} \geq 0, \quad \forall i \in [N],\ \mathbf{a} = (a_1, \ldots, a_N) \in \mathcal{A}.$$

Let $\bar{a}_i = \arg\max_{a_i} Q_i^P(h_i, a_i)$ for $i \in [N]$, and let $\bar{\mathbf{a}} = [\bar{a}_i]_{i \in [N]}$. Given that $\partial Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}) / \partial Q_1^P(h_1, a_1) \geq 0$ and $\bar{a}_1 = \arg\max_{a_1} Q_1^P(h_1, a_1)$, we have (for any $\mathbf{a} \in \mathcal{A}$)

$$Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}) \leq Q_{\text{tot}}^P(\mathbf{h}, \bar{a}_1, a_2, \ldots, a_N).$$

Applying the same logic to all $i \in [N]$ yields that for any $\mathbf{a} \in \mathcal{A}$,

$$\begin{aligned} Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}) &\leq Q_{\text{tot}}^P(\mathbf{h}, \bar{a}_1, \bar{a}_2, \ldots, \bar{a}_N) && (8.19) \\ &= Q_{\text{tot}}^P(\mathbf{h}, \bar{\mathbf{a}}). && (8.20) \end{aligned}$$

This implies that

$$\left(\arg\max_{a_1} Q_1^P(h_1, a_1), \ldots, \arg\max_{a_N} Q_N^P(h_N, a_N) \right) \subseteq \arg\max_{\mathbf{a}} Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}),$$

so $[Q_i^P]_{i \in [N]}$ satisfy IGM for Q_{tot}^P under $\mathbf{h}$. Therefore, by Theorem 10, $[Q_i^{\text{rob}}]_{i \in [N]}$ satisfy DrIGM for $Q_{\text{tot}}^{\mathcal{P}}$ under $\mathbf{h}$.

QTRAN condition. Given a joint history $\mathbf{h} \in \mathcal{H}$, for any $P \in \mathcal{P}$, we have (for all $\mathbf{a} = (a_1, \ldots, a_N) \in \mathcal{A}$):

$$\sum_{i=1}^{N} Q_i^P(h_i, a_i) - Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}) + V_{\text{tot}}(\mathbf{h}) = \begin{cases} 0, & \mathbf{a} = \bar{\mathbf{a}}, \\ \geq 0, & \mathbf{a} \neq \bar{\mathbf{a}}, \end{cases} \tag{8.21}$$

where $\bar{\mathbf{a}} = [\bar{a}_i]_{i\in[N]}$ with $\bar{a}_i = \arg\max_{a_i} Q_i^P(h_i, a_i)$ and $V_{\text{tot}}(\mathbf{h}) = \max_{\mathbf{a}} Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}) - \sum_{i=1}^{N} Q_i^P(h_i, a_i)$. Therefore,

$$\begin{aligned} Q_{\text{tot}}^P(\mathbf{h}, \bar{\mathbf{a}}) &= \sum_{i=1}^{N} Q_i^P(h_i, \bar{a}_i) + V_{\text{tot}}(\mathbf{h}) && \text{(eq. (8.21))} \\ &= \max_{\mathbf{a}} Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}), && \text{(Definition of } V(\mathbf{h})\text{)} \end{aligned}$$

This implies that

$$\left(\arg\max_{a_1} Q_1^P(h_1, a_1), \ldots, \arg\max_{a_N} Q_N^P(h_N, a_N)\right) \subseteq \arg\max_{\mathbf{a}} Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}),$$

so $[Q_i^P]_{i\in[N]}$ satisfy IGM for Q_{tot}^P under $\mathbf{h}$. Therefore, by Theorem 10, $[Q_i^{\text{rob}}]_{i\in[N]}$ satisfy DrIGM for $Q_{\text{tot}}^{\mathcal{P}}$ under $\mathbf{h}$.

Combining the three cases concludes the proof of Theorem 11. □

Proof of Theorem 12

Proof. Recall that given an uncertainty set $\mathcal{P}$, the robust joint action value is defined as,

$$Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h}, \mathbf{a}) := \inf_{P\in\mathcal{P}} Q_{\text{tot}}^P(\mathbf{h}, \mathbf{a}), \ \forall (\mathbf{h}, \mathbf{a}) \in \mathcal{H} \times \mathcal{A}. \tag{8.22}$$

Given that $P_{\text{test}} \in \mathcal{P}$, we directly have:

$$Q_{\text{tot}}^{\mathcal{P}}(\mathbf{h}, \mathbf{a}) \leq Q_{\text{tot}}^{P_{\text{test}}}(\mathbf{h}, \mathbf{a}), \ \forall \mathbf{h} \in \mathcal{H}, \ \mathbf{a} \in \mathcal{A}. \tag{8.23}$$

This concludes the proof. □

8.10 Robust Bellman Operators

We start by introducing the assumptions needed to derive the robust bellman operators.

Assumption 12 (Fail-state [198])**.** *The robust Dec-POMDP has a* fail state s_f *such that*

$$r(s_f, \mathbf{a}) = 0 \quad \textit{and} \quad P_{s_f,\mathbf{a}}(s_f) = 1, \qquad \forall \mathbf{a} \in \mathcal{A}, \ \forall P \in \mathcal{P}. \tag{8.24}$$

This requirement is mild, as fail states naturally arise in both simulated and physical systems. For example, in robotics, a configuration where the robot falls and cannot recover, whether in simulators such as MuJoCo or in real hardware, serves as a natural fail state. We can further relax it to the following assumption.

Assumption 13 (Vanishing minimal value [168])**.** *The underlying RMDP satisfies*

$$\min_{s\in\mathcal{S}} V^{\mathcal{P}}_{\text{tot}}(s) = 0. \tag{8.25}$$

Without loss of generality, we also assume that any initial state $s_1 \notin \arg\min_{s\in\mathcal{S}} V^{\mathcal{P}}_{\text{tot}}(s)$.

This assumption states that the lowest achievable robust value across all states is normalized to zero. The exclusion of the minimizing state as the starting point rules out the degenerate case where the agent begins with zero guaranteed return.

ρ-contamination uncertainty set. Given a ρ-contamination uncertainty set defined in eq. (8.7), the robust bellman operator can be expanded as:

$$(\mathcal{T}^{\mathcal{P}}Q)(\mathbf{h},\mathbf{a}) = r(s,\mathbf{a}) + \gamma \inf_{P_{\mathbf{h},\mathbf{a}}\in\mathcal{P}_{\mathbf{h},\mathbf{a}}} \mathbb{E}_{\mathbf{h}'\sim P_{\mathbf{h},\mathbf{a}}}\left[\max_{\mathbf{a}'\in\mathcal{A}} Q(\mathbf{h}',\mathbf{a}')\right]. \tag{8.26}$$

$$= r(s,\mathbf{a}) + \gamma(1-\rho)\,\mathbb{E}_{\mathbf{h}'\sim P^0_{\mathbf{h},\mathbf{a}}}\left[\max_{\mathbf{a}'\in\mathcal{A}} Q(\mathbf{h}',\mathbf{a}')\right] \tag{8.27}$$

$$+ \rho \min_{s'\in\mathcal{S}} V^{\mathcal{P}}_{\text{tot}}(s'). \tag{8.28}$$

Under assumption 12 (assumption 13), we obtain that:

$$(\mathcal{T}^{\mathcal{P}}Q)(\mathbf{h},\mathbf{a}) = r(s,\mathbf{a}) + \gamma(1-\rho)\,\mathbb{E}_{\mathbf{h}'\sim P^0_{\mathbf{h},\mathbf{a}}}\left[\max_{\mathbf{a}'\in\mathcal{A}} Q(\mathbf{h}',\mathbf{a}')\right]$$

(assumption 12 (assumption 13))

$$= r(s,\mathbf{a}) + \gamma(1-\rho)\,\mathbb{E}_{\mathbf{h}'\sim P^0_{\mathbf{h},\mathbf{a}}}\left[Q(\mathbf{h}',\bar{\mathbf{a}}')\right], \qquad \text{(definition 8)}$$

where $\bar{a}'_i = \arg\max_{a'_i} Q^{\text{rob}}_i(h'_i, a'_i)$.

TV-uncertainty set. Leveraging Panaganti et al. [198, Proposition 1], given a TV-uncertainty set defined in eq. (8.9), the robust bellman operator can be expanded as:

$$(\mathcal{T}Q^{\mathcal{P}}_{\text{tot}})(\mathbf{h},\mathbf{a}) = r(s,\mathbf{a}) - \inf_{\eta\in[0,\frac{2}{\rho(1-\gamma)}]} \gamma\,\mathbb{E}_{\mathbf{h}'\sim P^0_{\mathbf{h},\mathbf{a}}}\left(\rho\left[\eta(s,\mathbf{a}) - \inf_{s''\in\mathcal{S}} V^{\mathcal{P}}_{\text{tot}}(s'')\right]_+ - \eta\right.$$

$$\left. + \left[\eta(s,\mathbf{a}) - \max_{\mathbf{a}'\in\mathcal{A}} Q^{\mathcal{P}}_{\text{tot}}(\mathbf{h}',\mathbf{a}')\right]_+\right). \tag{8.29}$$

Under assumption 12 (assumption 13), we obtain that:

$$(\mathcal{T}Q^{\mathcal{P}}_{\text{tot}})(\mathbf{h},\mathbf{a}) = r(s,\mathbf{a}) - \inf_{\eta\in[0,\frac{2}{\rho(1-\gamma)}]} \gamma\,\mathbb{E}_{\mathbf{h}'\sim P^0_{\mathbf{h},\mathbf{a}}}\Bigg(-(1-\rho)\eta(s,\mathbf{a}) + \Big[\eta(s,\mathbf{a}) - \max_{\mathbf{a}'\in\mathcal{A}} Q^{\mathcal{P}}_{\text{tot}}(\mathbf{h}',\mathbf{a}')\Big]_+\Bigg).$$

(assumption 12 (assumption 13))

$$= r(s,\mathbf{a}) - \inf_{\eta\in[0,\frac{2}{\rho(1-\gamma)}]} \gamma\,\mathbb{E}_{\mathbf{h}'\sim P^0_{\mathbf{h},\mathbf{a}}}\Bigg(-(1-\rho)\eta(s,\mathbf{a}) + \Big[\eta(s,\mathbf{a}) - \max_{\mathbf{a}'\in\mathcal{A}} Q^{\mathcal{P}}_{\text{tot}}(h'_1,\ldots,h'_N,\bar{a}'_1,\ldots,\bar{a}'_N)\Big]_+\Bigg).$$

(definition 8)

8.11 Algorithms

We offer a full description of our algorithms in this section, presented in algorithms 8 to 13.

Algorithm 8 Robust VDN with ρ-contamination uncertainty set

Input robustness parameter ρ, target network update frequency f and ε
Initialize replay buffer $\mathcal{D}$
Initialize $[Q^{\text{rob}}_i]_{i\in[N]}$ with random parameters θ
Initialize target parameters $\theta^- = \theta$
for episode $e = 1,\ldots,E$ **do**
    Observe initial state s^0 and observation $o^0_i = \sigma_i(s^0)$ for each agent i.
    for $t = 1,\ldots,T$ **do**
        Each agent i chooses its action a^t_i using ε-greedy policy.
        Take joint action $\mathbf{a}^t$, observe the next state s^{t+1}, reward r^t and observation $o^{t+1}_i = \sigma_i(s^{t+1})$ for each agent i
        Store transition $(\mathbf{h}^t,\mathbf{a}^t,r^t,\mathbf{h}^{t+1})$ in replay buffer $\mathcal{D}$
        Sample a mini-batch of transitions $(\mathbf{h},\mathbf{a},r,\mathbf{h}')$ from $\mathcal{D}$
        Set $\bar{\mathbf{a}}' = [\arg\max_{a'_i} Q^{\text{rob}}_i(h'_i,a'_i;\theta^-)]_{i\in[N]}$
        Set $Q^{\mathcal{P},\text{VDN}}_{\text{tot}}(\mathbf{h},\mathbf{a};\theta) = \sum_{i\in[N]} Q^{\text{rob}}_i(h_i,a_i;\theta)$
        Set $Q^{\mathcal{P},\text{VDN}}_{\text{tot}}(\mathbf{h}',\bar{\mathbf{a}}';\theta^-) = \sum_{i\in[N]} Q^{\text{rob}}_i(h'_i,\bar{a}'_i;\theta^-)$
        Set $y^{\text{target}} = r + \gamma(1-\rho)Q^{\mathcal{P},\text{VDN}}_{\text{tot}}(\mathbf{h}',\bar{\mathbf{a}}';\theta^-)$
        Calculate TD loss $L_{\text{TD}} = (Q^{\mathcal{P},\text{VDN}}_{\text{tot}}(\mathbf{h},\mathbf{a};\theta) - y^{\text{target}})^2$
        Update θ by minimizing L_{TD}
        Update $\theta^- = \theta$ with frequency f

Algorithm 9 Robust QMIX with ρ-contamination uncertainty set

Input robustness parameter ρ, target network update frequency f and ε
Initialize replay buffer $\mathcal{D}$
Initialize $[Q_i^{\text{rob}}]_{i\in[N]}$ and mixing network f_θ with random parameters θ
Initialize target parameters $\theta^- = \theta$
for episode $e = 1, \ldots, E$ **do**
    Observe initial state s^0 and observation $o_i^0 = \sigma_i(s^0)$ for each agent i.
    for $t = 1, \ldots, T$ **do**
        Each agent i chooses its action a_i^t using ε-greedy policy.
        Take joint action $\mathbf{a}^t$, observe the next state s^{t+1}, reward r^t and observation $o_i^{t+1} = \sigma_i(s^{t+1})$ for each agent i
        Store transition $(\mathbf{h}^t, \mathbf{a}^t, s^t, r^t, \mathbf{h}^{t+1}, s^{t+1})$ in replay buffer $\mathcal{D}$
        Sample a mini-batch of transitions $(\mathbf{h}, \mathbf{a}, s, r, \mathbf{h}', s')$ from $\mathcal{D}$
        Set $\bar{\mathbf{a}}' = [\arg\max_{a_i'} Q_i^{\text{rob}}(h_i', a_i'; \theta^-)]_{i\in[N]}$
        Set $Q_{\text{tot}}^{\mathcal{P},\text{QMIX}}(\mathbf{h}, \mathbf{a}; \theta) = f_\theta((Q_i^{\text{rob}}(h_i, a_i; \theta))_{i\in[N]}, s)$
        Set $Q_{\text{tot}}^{\mathcal{P},\text{QMIX}}(\mathbf{h}', \bar{\mathbf{a}}'; \theta^-) = f_{\theta^-}((Q_i^{\text{rob}}(h_i', \bar{a}_i'; \theta^-))_{i\in[N]}, s')$
        Set $y^{\text{target}} = r + \gamma(1-\rho)Q_{\text{tot}}^{\mathcal{P},\text{QMIX}}(\mathbf{h}', \bar{\mathbf{a}}'; \theta^-)$
        Calculate TD loss $L_{\text{TD}} = (Q_{\text{tot}}^{\mathcal{P},\text{QMIX}}(\mathbf{h}, \mathbf{a}; \theta) - y^{\text{target}})^2$
        Update θ by minimizing L_{TD}
        Update $\theta^- = \theta$ with frequency f

8.12 Experiment details

Task Description

We test our algorithms and baseline algorithms in `BuildingEnv` in [310]. This environment considers the control of the heat flow in a multi-zone building so as to maintain a desired temperature setpoint. Building temperature simulation uses f*irst-principled physics models*, to capture the real-world dynamics. The environmental model and reward functions can differ from three climate types and locations (San Diego, Tucson, New York), which jointly decide the climate.

Episode. In `BuildingEnv`, each episode runs for 1 day, with 5-minute time intervals. That is, the horizon length $H = 288$, and the time interval length $\tau = 5/60$ hours. We set the discount factor $\gamma \simeq 0.997$ by using H as the effective horizon length $H = \frac{1}{1-\gamma}$.

Algorithm 10 Robust QTRAN with ρ-contamination uncertainty set

Input robustness parameter ρ, target network update frequency f and ε
Initialize replay buffer $\mathcal{D}$
Initialize $[Q_i^{\text{rob}}]_{i\in[N]}$, $Q_{\text{tot}}^{\mathcal{P},\text{QTRAN}}$ and $V_{\text{tot}}^{\mathcal{P},\text{QTRAN}}$ with random parameters θ
Initialize target parameters $\theta^- = \theta$
for episode $e = 1, \ldots, E$ **do**
    Observe initial state s^0 and observation $o_i^0 = \sigma_i(s^0)$ for each agent i.
    for $t = 1, \ldots, T$ **do**
        Each agent i chooses its action a_i^t using ε-greedy policy.
        Take joint action $\mathbf{a}^t$, observe the next state s^{t+1}, reward r^t and observation $o_i^{t+1} = \sigma_i(s^{t+1})$ for each agent i
        Store transition $(\mathbf{h}^t, \mathbf{a}^t, r^t, \mathbf{h}^{t+1})$ in replay buffer $\mathcal{D}$
        Sample a mini-batch of transitions $(\mathbf{h}, \mathbf{a}, r, \mathbf{h}')$ from $\mathcal{D}$
        Set $\bar{\mathbf{a}}' = [\arg\max_{a_i'} Q_i^{\text{rob}}(h_i', a_i'; \theta^-)]_{i\in[N]}$
        Set $y^{\text{target}} = r + \gamma(1-\rho)Q_{\text{tot}}^{\mathcal{P},\text{QTRAN}}(\mathbf{h}', \bar{\mathbf{a}}'; \theta^-)$
        Calculate TD loss $L_{\text{TD}} = (Q_{\text{tot}}^{\mathcal{P},\text{QTRAN}}(\mathbf{h}, \mathbf{a}; \theta) - y^{\text{target}})^2$
        Calculate L_{opt} using eq. (8.12)
        Calculate L_{nopt} using eq. (8.13)
        Update θ by minimizing $L = L_{\text{TD}} + L_{\text{opt}} + L_{\text{nopt}}$
        Update $\theta^- = \theta$ with frequency f

State Space. For a building with N indoor zones, the state contains observable properties of the building environment at timestep t:

$$s(t) = (T_1(t), \ldots, T_N(t), T_E(t), T_G(t), Q^{GHI}(t), \bar{Q}^p(t)), \tag{8.30}$$

where $T_i(t)$ denotes zone i's temperature at time step t, $\bar{Q}^p(t)$ is the heat acquisition from occupants' activities, $Q^{GHI}(t)$ is the heat gain from the solar irradiance, and $T_G(t)$ and $T_E(t)$ denote the ground and outdoor environment temperature.

Observation Space. For agent i, given the current state $s(t)$, its observation $o_i(t)$ is given by:

$$\begin{aligned} o_i(t) &= \sigma_i(s(t)) \\ &= (T_i(t), T_E(t), T_G(t), Q^{GHI}(t), \bar{Q}^p(t)). \end{aligned} \tag{8.31}$$

That is, agent i can observe the tempature at zone i, the heat acquisition from occupants' activities, the heat gain from the solar irradiance, and the ground and outdoor environment temperature.

Algorithm 11 Robust VDN with TV uncertainty set

Input robustness parameter ρ, target network update frequency f and ε
Initialize replay buffer $\mathcal{D}$
Initialize $[Q_i^{\text{rob}}]_{i\in[N]}$ with random parameters θ
Initialize dual network η_ξ with random parameters ξ
Initialize target parameters $\theta^- = \theta$
for episode $e = 1, \ldots, E$ **do**
    Observe initial state s^0 and observation $o_i^0 = \sigma_i(s^0)$ for each agent i.
    for $t = 1, \ldots, T$ **do**
        Each agent i chooses its action a_i^t using ε-greedy policy.
        Take joint action $\mathbf{a}^t$, observe the next state s^{t+1}, reward r^t and observation $o_i^{t+1} = \sigma_i(s^{t+1})$ for each agent i
        Store transition $(\mathbf{h}^t, \mathbf{a}^t, s^t, r^t, \mathbf{h}^{t+1})$ in replay buffer $\mathcal{D}$
        Sample a mini-batch of transitions $(\mathbf{h}, \mathbf{a}, r, s, \mathbf{h}')$
        Calculate dual loss L_{dual} using eq. (8.17)
        Update ξ by minimizing L_{dual}
        Sample another mini-batch of transitions $(\mathbf{h}, \mathbf{a}, s, r, \mathbf{h}')$ from $\mathcal{D}$
        Set $\bar{\mathbf{a}}' = [\arg\max_{a_i'} Q_i^{\text{rob}}(h_i', a_i'; \theta^-)]_{i\in[N]}$
        Set $Q_{\text{tot}}^{\mathcal{P},\text{VDN}}(\mathbf{h}, \mathbf{a}; \theta) = \sum_{i\in[N]} Q_i^{\text{rob}}(h_i, a_i; \theta)$
        Set $Q_{\text{tot}}^{\mathcal{P},\text{VDN}}(\mathbf{h}', \bar{\mathbf{a}}'; \theta^-) = \sum_{i\in[N]} Q_i^{\text{rob}}(h_i', \bar{a}_i'; \theta^-)$
        Set $y^{\text{target}} = r + \gamma(1-\rho)\eta_\xi(s, \mathbf{a}) - \gamma[\eta_\xi(s, \mathbf{a}) - Q_{\text{tot}}^{\mathcal{P},\text{VDN}}(\mathbf{h}', \bar{\mathbf{a}}'; \theta^-)]_+$
        Calculate TD loss $L_{\text{TD}} = (Q_{\text{tot}}^{\mathcal{P},\text{VDN}}(\mathbf{h}, \mathbf{a}; \theta) - y^{\text{target}})^2$
        Update θ by minimizing L_{TD}
        Update $\theta^- = \theta$ with frequency f

Action Space. At time t, agent i's action is a scalar $a_i(t) \in [-1, 1]$, which sets the controlled heating supplied to zone i. The joint action $\mathbf{a}(t) = (a_1(t), \ldots, a_N(t))$

Reward Function. The objective is to reduce energy consumption while keeping the temperature within a given comfort range. Therefore, the reward function is a weighted average of these two goals:

$$r(t) = -(1-\beta)||\mathbf{a}(t)||_2 - \beta||T^{\text{target}} - T(t)||_2, \tag{8.32}$$

where $T^{\text{target}} = (T_1^{\text{target}}(t), \ldots, T_N^{\text{target}})$ are the target temperatures and $T(t) = (T_1(t), \ldots, T_N(t))$ are the actual zonal temperature, and $||\cdot||_2$ denote the ℓ_2 norm. We use the same default hyperparameter β across all experiments.

Algorithm 12 Robust QMIX with TV uncertainty set

Input robustness parameter ρ, target network update frequency f and ε
Initialize replay buffer $\mathcal{D}$
Initialize $[Q_i^{\text{rob}}]_{i\in[N]}$ and mixing network f_θ with random parameters θ
Initialize dual network η_ξ with random parameters ξ
Initialize target parameters $\theta^- = \theta$
for episode $e = 1, \ldots, E$ **do**
    Observe initial state s^0 and observation $o_i^0 = \sigma_i(s^0)$ for each agent i.
    for $t = 1, \ldots, T$ **do**
        Each agent i chooses its action a_i^t using ε-greedy policy.
        Take joint action $\mathbf{a}^t$, observe the next state s^{t+1}, reward r^t and observation $o_i^{t+1} = \sigma_i(s^{t+1})$ for each agent i
        Store transition $(\mathbf{h}^t, \mathbf{a}^t, s^t, r^t, \mathbf{h}^{t+1})$ in replay buffer $\mathcal{D}$
        Sample a mini-batch of transitions $(\mathbf{h}, \mathbf{a}, r, \mathbf{h}')$
        Calculate dual loss L_{dual} using eq. (8.17)
        Update ξ by minimizing L_{dual}
        Sample another mini-batch of transitions $(\mathbf{h}, \mathbf{a}, s, r, \mathbf{h}')$ from $\mathcal{D}$
        Set $\bar{\mathbf{a}}' = [\arg\max_{a_i'} Q_i^{\text{rob}}(h_i', a_i'; \theta^-)]_{i\in[N]}$
        Set $Q_{\text{tot}}^{\mathcal{P},\text{QMIX}}(\mathbf{h}, \mathbf{a}; \theta) = f_\theta((Q_i^{\text{rob}}(h_i, a_i; \theta))_{i\in[N]}, s)$
        Set $Q_{\text{tot}}^{\mathcal{P},\text{QMIX}}(\mathbf{h}', \bar{\mathbf{a}}'; \theta^-) = f_{\theta^-}((Q_i^{\text{rob}}(h_i', \bar{a}_i'; \theta^-))_{i\in[N]}, s')$
        Set $y^{\text{target}} = r + \gamma(1-\rho)\eta_\xi(s, \mathbf{a}) - \gamma[\eta_\xi(s, \mathbf{a}) - Q_{\text{tot}}^{\mathcal{P},\text{QMIX}}(\mathbf{h}', \bar{\mathbf{a}}'; \theta^-)]_+$
        Calculate TD loss $L_{\text{TD}} = (Q_{\text{tot}}^{\mathcal{P},\text{QMIX}}(\mathbf{h}, \mathbf{a}; \theta) - y^{\text{target}})^2$
        Update θ by minimizing L_{TD}
        Update $\theta^- = \theta$ with frequency f

Environmental Uncertainty. The environmental model and reward functions can differ from three climate types and locations (San Diego, Tucson, New York), which jointly decide the climate. Besides, `BuildingEnv` contains distribution shifts in the ambient outdoor temperature profile T_E incurred by seasonal shifts.

Experiments Setup.

We implement distributionally robust algorithms with two types of uncertainty sets (ρ-contamination [321], TV-uncertainty[198]) and three different value factorization methods (VDN [259], QMIX [217], QTRAN [251]). We also implement the GroupDR MARL algorithm from [165], also with these three different value factorization methods. Each experiment is run independently with 5 different random seeds.

Algorithm 13 Robust QTRAN with TV uncertainty set

Input robustness parameter ρ, target network update frequency f and ε
Initialize replay buffer $\mathcal{D}$
Initialize $[Q_i^{\text{rob}}]_{i\in[N]}$, $Q_{\text{tot}}^{\mathcal{P},\text{QTRAN}}$ and $V_{\text{tot}}^{\mathcal{P},\text{QTRAN}}$ with random parameters θ
Initialize dual network η_ξ with random parameters ξ
Initialize target parameters $\theta^- = \theta$
for episode $e = 1, \ldots, E$ **do**
    Observe initial state s^0 and observation $o_i^0 = \sigma_i(s^0)$ for each agent i.
    for $t = 1, \ldots, T$ **do**
        Each agent i chooses its action a_i^t using ε-greedy policy.
        Take joint action $\mathbf{a}^t$, observe the next state s^{t+1}, reward r^t and observation $o_i^{t+1} = \sigma_i(s^{t+1})$ for each agent i
        Store transition $(\mathbf{h}^t, \mathbf{a}^t, s^t, r^t, \mathbf{h}^{t+1})$ in replay buffer $\mathcal{D}$
        Sample a mini-batch of transitions $(\mathbf{h}, \mathbf{a}, r, s, \mathbf{h}', s')$
        Calculate dual loss L_{dual} using eq. (8.17)
        Update ξ by minimizing L_{dual}
        Sample another mini-batch of transitions $(\mathbf{h}, \mathbf{a}, s, r, \mathbf{h}')$ from $\mathcal{D}$
        Set $\bar{\mathbf{a}}' = [\arg\max_{a_i'} Q_i^{\text{rob}}(h_i', a_i'; \theta^-)]_{i\in[N]}$
        Set $y^{\text{target}} = r + \gamma(1-\rho)\eta_\xi(s, \mathbf{a}) - \gamma[\eta_\xi(s, \mathbf{a}) - Q_{\text{tot}}^{\mathcal{P},\text{QTRAN}}(\mathbf{h}', \bar{\mathbf{a}}'; \theta^-)]_+$
        Calculate TD loss $L_{\text{TD}} = (Q_{\text{tot}}^{\mathcal{P},\text{QTRAN}}(\mathbf{h}, \mathbf{a}; \theta) - y^{\text{target}})^2$
        Calculate L_{opt} using eq. (8.12)
        Calculate L_{nopt} using eq. (8.13)
        Update θ by minimizing $L = L_{\text{TD}} + L_{\text{opt}} + L_{\text{nopt}}$
        Update $\theta^- = \theta$ with frequency f

Hyperparameters. Training lasts 600 episodes with parameter updates every 2 steps and target updates every 25k steps. Replay buffer size is 10k; batch size is 64. We use ε-greedy exploration with ε annealed from 1.0 to 0.01 over 120k steps. Hidden layer size is 64.

Environment configurations. The environment configurations we use throughout the three experiments are shown in Table 8.3 where `Env_1` is the training environment, and the other environments are numbered in order of how much they differ from `Env_1`.

Robust individual Q-networks. Each agent employs a recurrent Q-network [105], encoding its local observation concatenated with the previous action (one-hot). Features pass through a fully connected layer with ReLU, followed by a single-layer

Environment	Weather	Location
`Env_1`	Hot_Dry	Tucson
`Env_2`	Hot_Humid	Tampa
`Env_3`	Very_Hot_Humid	Honolulu
`Env_4`	Warm_Dry	El Paso
`Env_5`	Cool_Marine	Seattle
`Env_6`	Mixed_Humid	New York

Table 8.3: Environment configurations for SustainGym `BuildingEnv`.

LSTM (hidden size 64). The final hidden state is projected into Q-values. We optimize with RMSprop (lr = 5×10^{-4}).

Mixing networks (QMIX). Following TorchRL [40], we use hypernetworks to generate state-dependent mixing weights while enforcing monotonicity. Per-agent Q-values are embedded, passed through ELU, and projected to a scalar joint Q. A state-conditioned bias is added via a two-layer MLP. Optimizer: RMSprop (lr = 5×10^{-4}).

QTRAN Networks. We implement joint action-value and state-value networks as in Son et al. [251]. The joint action is encoded by concatenated one-hots, projected with ReLU, and optionally combined with agent features. The state-value network processes the global state and summed agent features in parallel, concatenates them, and outputs a scalar. Both use Adam (lr = 10^{-3}).

Dual networks (TV uncertainty). The global state and joint action (concatenated one-hots) are separately embedded via ReLU MLPs, concatenated, and passed through another ReLU layer before a final linear projection to a scalar. Optimizer: Adam (lr = 10^{-3}).

GroupDR training. GroupDR first fits a contextual-bandit-based worst-case reward estimator to estimate the worst-case reward, using data from `Env_1` to `Env_5` collected under a VDN-trained behavior policy. Using this estimator, we then train the individual Q-networks by randomly sampling episodes from `Env_1` to `Env_5`. (For fairness, `Env_6` is never used for training by any method.)

Normalized Return. Because the reward in `BuildingEnv` is negative, we normalize the returns to $[0, 1]$ by the following transformation:

$$\text{Normalized\_return} = \frac{\text{Return} + 9000}{9000}. \tag{8.33}$$

Experiment 1: climatic shifts. During training, we train each algorithm in `Env_1` by sampling episodes from Day 1 to Day 200. During evaluation, we evaluate each algorithm in `Env_1` to `Env_6`, by sampling 50 episodes from Day 1 to Day 200 in each environment. This setup demonstrates distribution shift induced by climate differences.

Experiment 2: seasonal shifts. During training, we train each algorithm in `Env_1` by sampling episodes from Day 1 to Day 200. During evaluation, we evaluate each algorithm in `Env_1`, by sampling 50 episodes from Day 400 to Day 600 in each environment. This setup demonstrates distribution shift induced by seasonality within the same location.

Experiment 3: climatic and seasonal shifts. During training, we train each algorithm in `Env_1` by sampling episodes from Day 1 to Day 200. During evaluation, we evaluate each algorithm in `Env_6`, by sampling 50 episodes from Day 400 to Day 600 in each environment. This setup demonstrates the combined effects of climate and seasonal shifts.

Chapter 9

CONCLUSIONS AND FUTURE DIRECTIONS

9.1 Conclusion

This thesis has studied how uncertainty representations can be learned, calibrated, and optimized in service of downstream decisions. Across the batch and online settings, the central message has been consistent: uncertainty should not be treated as a passive byproduct of prediction. Instead, the uncertainty object itself – whether a scalar risk threshold, a prediction set, a learned ambiguity set, or a robust value function – should be designed with the decision problem in mind.

In the batch setting, the thesis developed several end-to-end methods for learning decision-focused uncertainty representations. Conformal risk training (Chapter 2) extended conformal risk control from expected losses to tail-risk objectives such as CVaR, enabling predictors to be trained jointly with a risk-controlling procedure. Chapter 3 then moved from scalar risk control to set-valued uncertainty, showing how conformally calibrated prediction sets can be learned end-to-end for conditional robust optimization. Gen-WDRO (Chapter 4) further generalized this idea by learning conditional distributions and coupling them to a Wasserstein distributionally robust optimization layer, while diffusion-based decision-focused learning (Chapter 5) showed that rich generative models can support stochastic optimization through end-to-end training. Taken together, these chapters demonstrate that uncertainty quantification and decision-making need not be separate stages.

The online part of the thesis explored how uncertainty can support sequential control and reinforcement learning in realistic sustainability applications. Chapter 6 established a finite-mistake guarantee for online voltage control under unknown grid topology, showing that safety and learning can be combined in a provable way. SustainGym (Chapter 7) provided a benchmark suite that exposes distribution shift, physical constraints, and multi-agent interaction in sustainability settings, making it easier to evaluate RL methods in environments that resemble real deployment challenges. Finally, DrIGM (Chapter 8) extended the individual-global-maximum principle to distributionally robust multi-agent RL, enabling decentralized greedy execution under uncertainty while preserving robust value alignment. These results suggest that decision-focused uncertainty representations are equally valuable in

sequential settings, where they can improve both stability and generalization.

9.2 Future Directions

While this thesis has established the importance of decision-focused uncertainty representations, there are many exciting directions for future work that build on the core idea of learning such representations.

Online conformal tail risk control. An important extension of conformal CVaR control is to the online, adversarial setting. In this regime, the learner would face a sequence of losses or constraints that may vary over time, and the goal would be to make decisions with provable regret guarantees rather than only batch risk-control guarantees. A natural target is sublinear regret for both the objective and the risk constraint, which would provide an online analogue of the finite-sample guarantees developed in the batch setting. Achieving this would require combining ideas from online learning, adaptive calibration, and risk-sensitive optimization in a way that remains tractable under minimal assumptions.

Decision-focused learning for non-convex problems. Most of the decision-focused learning methods developed in this thesis rely on convex or otherwise structured downstream optimization problems, where differentiation through the decision layer can be handled analytically or with efficient implicit methods. Many applications of interest, however, are non-convex. Extending decision-focused learning to this regime would substantially broaden its scope. One promising direction is to leverage policy-gradient methods, including deterministic policy gradients, to optimize decision-focused objectives when the downstream problem is not amenable to convex differentiation. A key challenge will be to preserve the decision-awareness of the training objective while managing the additional optimization instability introduced by non-convexity.

Using learned uncertainty more directly in RL. The DrIGM framework demonstrates that learned uncertainty can be incorporated into RL through robust value functions that preserve decentralized execution. More generally, there may be several other ways to inject learned uncertainty into RL algorithms. One possibility is to use uncertainty estimates to guide exploration, rather than only to robustify evaluation or policy execution. Another is to couple uncertainty-aware representations more tightly with model-based planning or offline RL, where distribution shift and extrapolation

error are central concerns. Exploring these directions could yield RL algorithms that are not only more robust, but also more sample-efficient and better aligned with real-world risk management.

Real-world deployment in sustainability and energy systems. Finally, one direction that this thesis did not explore in depth is the real-world deployment of the methods developed here. While the theoretical and empirical results are promising, there are many practical challenges to deploying these methods in operational sustainability systems, including issues of scalability, interpretability, and integration with existing control frameworks. Future work that bridges the gap between research and practice, perhaps through partnerships with industry or utilities, could help to validate and refine these methods in real-world settings, ultimately contributing to more sustainable and resilient energy systems.

The broader lesson of the thesis is that uncertainty representations are most useful when they are learned with the decision in mind. Future work that continues to unify statistical guarantees, optimization, and control is likely to be especially valuable in safety-critical sustainability systems, where the costs of miscalibrated uncertainty can be substantial.

BIBLIOGRAPHY

[1] Heba M. Abdullah, Adel Gastli, and Lazhar Ben-Brahim. 2021. Reinforcement Learning Based EV Charging Management Systems–A Review. *IEEE Access*, 9, 41506–41531. doi:10.1109/ACCESS.2021.3064354.

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2020. An Optimistic Perspective on Offline Reinforcement Learning. en. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, (Nov. 2020), 104–114. Retrieved Nov. 29, 2025 from https://proceedings.mlr.press/v119/agarwal20c.html.

[3] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. 2019. Differentiable Convex Optimization Layers. In *Advances in Neural Information Processing Systems*. H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, (Eds.) Vol. 32. Curran Associates, Inc. https://papers.nips.cc/paper_files/paper/2019/hash/9ce3c52fc54362e22053399d3181c638-Abstract.html.

[4] Akshay Agrawal, Shane Barratt, Stephen Boyd, and Walaa M. Moursi. 2019. Differentiating through a cone program. en-US. *Journal of Applied and Numerical Optimization*, 1, 2, (Aug. 2019), 107–115. doi:10.23952/jano.1.2019.2.02.

[5] Polina Alexeenko and Eilyan Bitar. 2020. Nonparametric Estimation of Uncertainty Sets for Robust Optimization. en. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, Jeju, Korea (South), (Dec. 2020), 1196–1203. ISBN: 978-1-72817-447-1. doi:10.1109/CDC42340.2020.9303863.

[6] Saud Alghumayjan, Bolun Xu, and Ming Yi. 2025. Conformal Uncertainty Quantification of Electricity Price Predictions for Risk-Averse Storage Arbitrage. In *2025 IEEE Power & Energy Society General Meeting (PESGM)*. ISSN: 1944-9933. (July 2025), 1–5. doi:10.1109/PESGM52009.2025.11225098.

[7] Brandon Amos and J. Zico Kolter. 2017. OptNet: Differentiable Optimization as a Layer in Neural Networks. en. In *Proceedings of the 34th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, (July 2017), 136–145. Retrieved Mar. 29, 2023 from https://proceedings.mlr.press/v70/amos17a.html.

[8] Brandon Amos, Lei Xu, and J. Zico Kolter. 2017. Input Convex Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning* (Proceedings of Machine Learning Research). Vol. 70. PMLR, (Aug. 2017), 146–155. Retrieved June 20, 2022 from https://proceedings.mlr.press/v70/amos17b.html.

[9] Anastasios N. Angelopoulos and Stephen Bates. 2023. Conformal Prediction: A Gentle Introduction. *Foundations and Trends® in Machine Learning*, 16, 4, 494–591. doi:10.1561/2200000101.

[10] Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. 2025. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19, 2, (June 2025), 1641–1662. doi:10.1214/24-AOAS1998.

[11] Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. 2023. Conformal Risk Control. en. arXiv:2208.02814 [cs, math, stat]. (Apr. 2023). Retrieved Oct. 16, 2024 from http://arxiv.org/abs/2208.02814.

[12] Omid Ardakanian, Vincent WS Wong, Roel Dobbe, Steven H Low, Alexandra von Meier, Claire J Tomlin, and Ye Yuan. 2019. On identification of distribution grids. *IEEE Trans. on Control Netw. Syst.*, 6, 3, (Sept. 2019), 950–960.

[13] CJ Argue, Sébastien Bubeck, Michael B Cohen, Anupam Gupta, and Yin Tat Lee. 2019. A nearly-linear bound for chasing nested convex bodies. In *Proc. 30th Annu. ACM-SIAM Symp. Discrete Algorithms*, 117–122.

[14] Yossi Arjevani, Yair Carmon, John C. Duchi, Dylan J. Foster, Ayush Sekhari, and Karthik Sridharan. 2020. Second-Order Information in Non-Convex Stochastic Optimization: Power and Limitations. en. In *Proceedings of Twenty Second Conference on Learning Theory*. PMLR, (July 2020), 242–299. Retrieved Aug. 7, 2025 from https://proceedings.mlr.press/v125/arjevani20a.html.

[15] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. 1999. Coherent Measures of Risk. en. *Mathematical Finance*, 9, 3, (July 1999), 203–228. doi:10.1111/1467-9965.00068.

[16] Bharathan Balaji et al. 2019. Deepracer: educational autonomous racing platform for experimentation with sim2real reinforcement learning. *arXiv preprint arXiv:1911.01562*. arXiv: 1911.01562 [cs.LG].

[17] Mesut E Baran and Felix F Wu. 1989. Optimal capacitor placement on radial distribution systems. *IEEE Trans. Power Del.*, 4, 1, 725–734.

[18] Mesut E Baran and Felix F Wu. 1989. Optimal sizing of capacitors placed on a radial distribution system. *IEEE Trans. Power Del.*, 4, 1, 735–743. doi:10.1109/61.19266.

[19] Shane Barratt. 2019. On the Differentiability of the Solution to Convex Optimization Problems. arXiv:1804.05098 [math]. (Nov. 2019). doi:10.48550/arXiv.1804.05098.

[20] Clayton Barrows et al. 2020. The IEEE Reliability Test System: A Proposed 2019 Update. *IEEE Transactions on Power Systems*, 35, 1, (Jan. 2020), 119–127. doi:10.1109/TPWRS.2019.2925557.

[21] Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael Jordan. 2021. Distribution-free, Risk-controlling Prediction Sets. *Journal of the ACM*, 68, 6, (Sept. 2021), 43:1–43:34. doi:10.1145/3478535.

[22] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. 2009. *Robust Optimization*. Princeton University Press. ISBN: 978-0-691-14368-2. Retrieved Apr. 5, 2024 from https://www.jstor.org/stable/j.ctt7sk8p.

[23] Aharon Ben-Tal and Marc Teboulle. 2007. An Old-New Concept of Convex Risk Measures: The Optimized Certainty Equivalent. en. *Mathematical Finance*, 17, 3, 449–476. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9965.2007.00311.x. doi:10.1111/j.1467-9965.2007.00311.x.

[24] Aharon Ben-Tal and Marc Teboulle. 1986. Expected Utility, Penalty Functions, and Duality in Stochastic Nonlinear Programming. en. *Management Science*, 32, 11, (Nov. 1986), 1445–1466. doi:10.1287/mnsc.32.11.1445.

[25] J. Bernal, J. Sánchez, and F. Vilariño. 2012. Towards automatic polyp detection with a polyp appearance model. *Pattern Recognition*. Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011) 45, 9, (Sept. 2012), 3166–3182. doi:10.1016/j.patcog.2012.03.002.

[26] Jorge Bernal, F. Javier Sánchez, Gloria Fernández-Esparrach, Debora Gil, Cristina Rodríguez, and Fernando Vilariño. 2015. WM-DOVA maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians. *Computerized Medical Imaging and Graphics*, 43, (July 2015), 99–111. doi:10.1016/j.compmedimag.2015.02.007.

[27] Andrey Bernstein and Emiliano Dall'Anese. 2019. Real-time feedback-based optimization of distribution grids: a unified approach. *IEEE Trans. on Control of Netw. Syst.*, 6, 3, (Sept. 2019), 1197–1209.

[28] Dimitri P. Bertsekas, (Ed.) 2009. *Convex Optimization Theory*. en. Athena Scientific, Belmont, Mass. ISBN: 978-1-886529-31-1.

[29] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. 2018. Data-driven robust optimization. en. *Mathematical Programming*, 167, 2, (Feb. 2018), 235–292. doi:10.1007/s10107-017-1125-8.

[30] Dimitris Bertsimas and Nathan Kallus. 2020. From Predictive to Prescriptive Analytics. en. *Management Science*, 66, 3, (Mar. 2020), 1025–1044. doi:10.1287/mnsc.2018.3253.

[31] Dimitris Bertsimas, Eugene Litvinov, Xu Andy Sun, Jinye Zhao, and Tongxin Zheng. 2013. Adaptive Robust Optimization for the Security Constrained Unit Commitment Problem. *IEEE Transactions on Power Systems*, 28, 1, (Feb. 2013), 52–63. Conference Name: IEEE Transactions on Power Systems. doi:10.1109/TPWRS.2012.2205021.

[32] Dimitris Bertsimas and Aurélie Thiele. 2006. Robust and Data-Driven Optimization: Modern Decision Making Under Uncertainty. In *Models, Methods, and Applications for Innovative Decision Making*. INFORMS TutORials in Operations Research. Section: 4. INFORMS, (Sept. 2006), 95–122. ISBN: 978-1-877640-20-9. doi:10.1287/educ.1063.0022.

[33] D. Bessis and F. Clarke. 1999. Partial subdifferentials, derivates and Rademacher's Theorem. en. *Transactions of the American Mathematical Society*, 351, 7, 2899–2926. doi:10.1090/S0002-9947-99-02203-5.

[34] David Biagioni, Xiangyu Zhang, Dylan Wald, Deepthi Vaidhynathan, Rohit Chintala, Jennifer King, and Ahmed S. Zamzam. 2021. PowerGridworld: A Framework for Multi-Agent Reinforcement Learning in Power Systems, (Nov. 2021). doi:10.48550/arXiv.2111.05969.

[35] Jose Blanchet, Miao Lu, Tong Zhang, and Han Zhong. 2023. Double pessimism is provably efficient for distributionally robust offline reinforcement learning: generic algorithm and robust partial coverage. *Advances in Neural Information Processing Systems*, 36.

[36] Jose Blanchet and Karthyek Murthy. 2019. Quantifying distributional model risk via optimal transport. *Mathematics of Operations Research*, 44, 2, 565–600.

[37] Nicolas Blin. 2024. Accelerate Large Linear Programming Problems with NVIDIA cuOpt. en-US. (Oct. 2024). Retrieved Aug. 27, 2025 from https://developer.nvidia.com/blog/accelerate-large-linear-programming-problems-with-nvidia-cuopt/.

[38] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight Uncertainty in Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning* (Proceedings of Machine Learning Research). Vol. 37. PMLR, (May 2015), 1613–1622. Retrieved Aug. 5, 2022 from https://proceedings.mlr.press/v37/blundell15.

[39] Saverio Bolognani, Ruggero Carli, Guido Cavraro, and Sandro Zampieri. 2015. Distributed reactive power feedback control for voltage regulation and loss minimization. *IEEE Trans. Autom. Control*, 60, 4, (Apr. 2015), 966–981. doi:10.1109/TAC.2014.2363931.

[40] Albert Bou, Matteo Bettini, Sebastian Dittert, Vikash Kumar, Shagun Sodhani, Xiaomeng Yang, Gianni De Fabritiis, and Vincent Moens. 2023. Torchrl: a data-driven decision-making library for pytorch. *arXiv preprint arXiv:2306.00577*. arXiv: 2306.00577 [cs.LG].

[41] David Brandfonbrener, William Whitney, Rajesh Ranganath, and Joan Bruna. 2021. Offline Contextual Bandits with Overparameterized Models. en. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, (July 2021), 1049–1058. Retrieved Nov. 29, 2025 from https://proceedings.mlr.press/v139/brandfonbrener21a.html.

[42] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *arXiv:1606.01540 [cs]*, (June 2016). Retrieved Apr. 13, 2022 from http://arxiv.org/abs/1606.01540.

[43] Jean-Sebastien Brouillon, Emanuele Fabbiani, Pulkit Nahata, Keith Moffat, Florian Dörfler, and Giancarlo Ferrari-Trecate. 2023. Bayesian error-in-variables models for the identification of distribution grids. *IEEE Trans. Smart Grid*, 14, 2, (Mar. 2023), 1289–1299.

[44] Jean-Sébastien Brouillon, Keith Moffat, Florian Dörfler, and Giancarlo Ferrari-Trecate. 2022. Robust online joint state/input/parameter estimation of linear systems. In *Prof. IEEE 61st Conf. Decis. Control (CDC)*, 2153–2158.

[45] Tom Brown et al. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 1877–1901. Retrieved Jan. 22, 2026 from https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[46] Sébastien Bubeck, Bo'az Klartag, Yin Tat Lee, Yuanzhi Li, and Mark Sellke. 2020. Chasing nested convex bodies nearly optimally. In *Proc. 31st Annu. ACM-SIAM Symposium on Discrete Algorithms*, 1496–1508. ISBN: 9781611975994. doi:10.1137/1.9781611975994.91.

[47] Alexander Bukharin, Yan Li, Yue Yu, Qingru Zhang, Zhehui Chen, Simiao Zuo, Chao Zhang, Songan Zhang, and Tuo Zhao. 2023. Robust multi-agent reinforcement learning via adversarial regularization: theoretical foundation and stable algorithms. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[48] CAISO. 2016. What the duck curve tells us about managing a green grid. Tech. rep. California Independent System Operator, (Nov. 2016), 4.

[49] Jun Cao, Dan Harrold, Zhong Fan, Thomas Morstyn, David Healey, and Kang Li. 2020. Deep Reinforcement Learning-Based Energy Storage Arbitrage With Accurate Lithium-Ion Battery Degradation Model. *IEEE Transactions on Smart Grid*, 11, 5, (Sept. 2020), 4513–4521. doi:10.1109/TSG.2020.2986333.

[50] Jesus Capitan, Matthijs T.J. Spaan, Luis Merino, and Anibal Ollero. 2012. Decentralized multi-robot cooperation with auctioned pomdps. In *2012 IEEE International Conference on Robotics and Automation*, 3323–3328. doi:10.1109/ICRA.2012.6224917.

[51] Guido Cavraro and Vassilis Kekatos. 2018. Graph algorithms for topology identification using power grid probing. *IEEE Control Syst. Lett.*, 2, 4, 689–694.

[52] Catherine Yu-Chi Chen, Jingyan Shen, Zhun Deng, and Lihua Lei. 2025. Conformal Tail Risk Control for Large Language Model Alignment. arXiv:2502.20285 [cs]. (Feb. 2025). doi:10.48550/arXiv.2502.20285.

[53] Yize Chen, Yuanyuan Shi, and Baosen Zhang. 2020. Data-driven optimal voltage regulation using input convex neural networks. *Electr. Power Syst. Res.*, 189, (Dec. 2020). Art. no. 106741. doi:10.1016/j.epsr.2020.106741.

[54] Yize Chen, Yuanyuan Shi, and Baosen Zhang. 2019. Optimal Control Via Neural Networks: A Convex Approach. en. In *International Conference on Learning Representations*. Retrieved Aug. 22, 2023 from https://openreview.net/forum?id=H1MW72AcK7.

[55] Abhilash Reddy Chenreddy, Nymisha Bandi, and Erick Delage. 2022. Data-Driven Conditional Robust Optimization. en. In *Advances in Neural Information Processing Systems*. Vol. 35. (Dec. 2022), 9525–9537. Retrieved May 10, 2024 from https://proceedings.neurips.cc/paper_files/paper/2022/hash/3df874367ce2c43891aab1ab23ae6959-Abstract-Conference.html.

[56] Abhilash Reddy Chenreddy and Erick Delage. 2024. End-to-end Conditional Robust Optimization. en. In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*. PMLR, (Sept. 2024), 736–748. Retrieved Nov. 10, 2024 from https://proceedings.mlr.press/v244/chenreddy24a.html.

[57] Nicolas Christianson, Lucien Werner, Adam Wierman, and Steven Low. 2022. Dispatch-aware planning for feasible power system operation. *Electric Power Systems Research*, 212, (Nov. 2022), 108597. doi:10.1016/j.epsr.2022.108597.

[58] Nicolas Christianson, Christopher Yeh, Tongxin Li, Mahdi Torabi Rad, Azarang Golmohammadi, and Adam Wierman. 2022. Robustifying machine-learned algorithms for efficient grid operation. In *NeurIPS 2022 Workshop on Tackling Climate Change with Machine Learning*. https://www.climatechange.ai/papers/neurips2022/19.

[59] Alvaro H. Correia, Fabio V. Massoli, Christos Louizos, and Arash Behboodi. 2024. An Information Theoretic Perspective on Conformal Prediction. en. *Advances in Neural Information Processing Systems*, 37, (Dec. 2024), 101000–101041. Retrieved May 9, 2025 from https://proceedings.neurips.cc/paper_files/paper/2024/hash/b6fa3ed9624c184bd73e435123bd576a-Abstract-Conference.html.

[60] Santiago Cortes-Gomez, Carlos Miguel Patiño, Yewon Byun, Steven Wu, Eric Horvitz, and Bryan Wilder. 2024. Utility-Directed Conformal Prediction: A Decision-Aware Framework for Actionable Uncertainty Quantification. en. In (Oct. 2024). Retrieved May 16, 2025 from https://openreview.net/forum?id=iOMnn1hSBO.

[61] Giorgio Costa and Garud N. Iyengar. 2023. Distributionally robust end-to-end portfolio construction. *Quantitative Finance*, 23, 10, (Oct. 2023), 1465–1482. Publisher: Routledge _eprint: https://doi.org/10.1080/14697688.2023.2236148. doi:10.1080/14697688.2023.2236148.

[62] Wenqi Cui, Jiayi Li, and Baosen Zhang. 2022. Decentralized safe reinforcement learning for inverter-based voltage control. *Electr. Power Syst. Res.*, 211. Art no. 108609.

[63] Payman Dehghanian and Mladen Kezunovic. 2016. Probabilistic impact of transmission line switching on power system operating states. In *Proc. IEEE/PES Transmiss. Distrib. Conf. Expo.* 1–5.

[64] Deepjyoti Deka, Scott Backhaus, and Michael Chertkov. 2016. Estimating distribution grid topologies: A graphical learning based approach. In *Prof. Power Syst. Comput. Conf.* 1–7. ISBN: 978-88-941051-2-4. doi:10.1109/PSCC.2016.7541005.

[65] Deepjyoti Deka, Scott Backhaus, and Michael Chertkov. 2018. Structure learning in power distribution networks. *IEEE Trans. Control Netw. Syst.*, 5, 3, (Sept. 2018), 1061–1074. doi:10.1109/TCNS.2017.2673546.

[66] Deepjyoti Deka, Vassilis Kekatos, and Guido Cavraro. 2024. Learning distribution grid topologies: a tutorial. *IEEE Trans. Smart Grid*, 15, 1, (Jan. 2024), 999–1013.

[67] Esther Derman, Daniel J Mankowitz, Timothy A Mann, and Shie Mannor. 2018. Soft-robust actor-critic policy-gradient. In *AUAI press for Association for Uncertainty in Artificial Intelligence*, 208–218.

[68] Esther Derman and Shie Mannor. 2020. Distributional robustness and regularization in reinforcement learning. *arXiv preprint arXiv:2003.02894*.

[69] Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 8780–8794. Retrieved Sept. 18, 2025 from https://proceedings.neurips.cc/paper_files/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

[70] Steven Diamond and Stephen Boyd. 2016. CVXPY: A python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.*, 17, 83, 2909–2913.

[71] US DOE. 2022. Engineering reference. https://energyplus.net/assets/nrel_custom/pdfs/pdfs_v22.1.0/EngineeringReference.pdf. (2022).

[72] Jing Dong, Jingwei Li, Baoxiang Wang, and Jingzhao Zhang. 2022. Online policy optimization for robust MDP. *arXiv preprint arXiv:2209.13841*.

[73] Priya L. Donti, Brandon Amos, and J. Zico Kolter. 2017. Task-based End-to-end Model Learning in Stochastic Optimization. In *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., Long Beach, CA, USA, (Dec. 2017). Retrieved Dec. 13, 2022 from http://papers.nips.cc/paper/7132-task-based-end-to-end-model-learning-in-stochastic-optimization.

[74] Jiajun Duan, Di Shi, Ruisheng Diao, Haifeng Li, Zhiwei Wang, Bei Zhang, Desong Bian, and Zhehan Yi. 2020. Deep-reinforcement-learning-based autonomous voltage control for power grid operations. *IEEE Trans. Power Syst.*, 35, 1, (Jan. 2020), 814–817. doi:10.1109/TPWRS.2019.2941134.

[75] John C. Duchi and Hongseok Namkoong. 2021. Learning models with uniform performance via distributionally robust optimization. en. *The Annals of Statistics*, 49, 3, (June 2021). doi:10.1214/20-AOS2004.

[76] Yury Dvorkin. 2020. A Chance-Constrained Stochastic Electricity Market. *IEEE Transactions on Power Systems*, 35, 4, (July 2020), 2993–3003. Conference Name: IEEE Transactions on Power Systems. doi:10.1109/TPWRS.2019.2961231.

[77] Bat-Sheva Einbinder, Yaniv Romano, Matteo Sesia, and Yanfei Zhou. 2022. Training Uncertainty-Aware Classifiers with Conformalized Deep Learning. In *Advances in Neural Information Processing Systems*. arXiv:2205.05878 [cs, stat]. New Orleans, LA, USA, (Nov. 2022). doi:10.48550/arXiv.2205.05878.

[78] Adam N. Elmachtoub and Paul Grigas. 2022. Smart "Predict, then Optimize". *Management Science*, 68, 1, (Jan. 2022), 9–26. Publisher: INFORMS. doi:10.1287/mnsc.2020.3922.

[79] Adam N. Elmachtoub, Henry Lam, Haofeng Zhang, and Yunfan Zhao. 2023. Estimate-Then-Optimize versus Integrated-Estimation-Optimization versus Sample Average Approximation: A Stochastic Dominance Perspective. arXiv:2304.06833 [cs, stat]. (Aug. 2023). doi:10.48550/arXiv.2304.06833.

[80] Saeid Esmaeili, Amjad Anvari-Moghaddam, Shahram Jadid, and Josep M Guerrero. 2019. Optimal simultaneous day-ahead scheduling and hourly reconfiguration of distribution systems considering responsive loads. *Int. J. Elect. Power Energy Syst.*, 104, 537–548.

[81] Emanuele Fabbiani, Pulkit Nahata, Giuseppe De Nicolao, and Giancarlo Ferrari-Trecate. 2021. Identification of AC distribution networks with recursive least squares and optimal design of experiment. *IEEE Trans. Control Syst. Technol.*, 30, 4, 1750–1757.

[82] Deng-Ping Fan, Ge-Peng Ji, Tao Zhou, Geng Chen, Huazhu Fu, Jianbing Shen, and Ling Shao. 2020. PraNet: Parallel Reverse Attention Network for Polyp Segmentation. arXiv:2006.11392 [eess]. (July 2020). doi:10.48550/arXiv.2006.11392.

[83] Masoud Farivar, Lijun Chen, and Steven Low. 2013. Equilibrium and dynamics of local voltage control in distribution systems. In *Proc. IEEE Conf. Decis. Control*. Firenze, Italy, 4329–4334. ISBN: 978-1-4673-5717-3. doi:10.1109/CDC.2013.6760555.

[84] Masoud Farivar, Russell Neal, Christopher Clarke, and Steven Low. 2012. Optimal inverter VAR control in distribution systems with high PV penetration. In *IEEE Power Eng. Soc. Gen. Meeting*. San Diego, CA, USA, 1–7. doi:10.1109/PESGM.2012.6345736.

[85] Jie Feng, Yuanyuan Shi, Guannan Qu, Steven H Low, Anima Anandkumar, and Adam Wierman. 2022. Stability constrained reinforcement learning for real-time voltage control in distribution systems. *arXiv:2209.07669*.

[86] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (NIPS'16). Curran Associates Inc., Barcelona, Spain, 2145–2153. ISBN: 9781510838819.

[87] Diego Fonseca and Mauricio Junca. 2023. Decision-dependent distributionally robust optimization. *arXiv preprint arXiv:2303.03971*.

[88] G. Foster and S. Rahmstorf. 2026. Global Warming Has Accelerated Significantly. en. *Geophysical Research Letters*, 53, 5. doi:10.1029/2025GL118804.

[89] Germano Gabbianelli, Gergely Neu, and Matteo Papini. 2024. Importance-Weighted Offline Learning Done Right. en. In *Proceedings of The 35th International Conference on Algorithmic Learning Theory*. PMLR, (Mar. 2024), 614–634. Retrieved Dec. 2, 2025 from https://proceedings.mlr.press/v237/gabbianelli24a.html.

[90] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *arXiv:1506.02142 [cs, stat]*. arXiv: 1506.02142. (Oct. 2016). Retrieved May 10, 2022 from http://arxiv.org/abs/1506.02142.

[91] Chao Gao and Miles A Redfern. 2010. A review of voltage control techniques of networks with distributed generations using on-load tap changer transformers. In *Proc. 45th Int. Univ. Power Eng. Conf. (UPEC)*. Cardiff, UK, 1–6.

[92] Rui Gao. 2020. Finite-sample guarantees for wasserstein distributionally robust optimization: breaking the curse of dimensionality. *arXiv preprint arXiv:2009.04382*.

[93] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. 2021. Res2Net: A New Multi-Scale Backbone Architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43, 2, (Feb. 2021), 652–662. doi:10.1109/TPAMI.2019.2938758.

[94] Yuanqi Gao, Wei Wang, and Nanpeng Yu. 2021. Consensus Multi-Agent Reinforcement Learning for Volt-VAR Control in Power Distribution Networks. *IEEE Transactions on Smart Grid*, 12, 4, (July 2021), 3594–3604. doi:10.1109/TSG.2021.3058996.

[95] 2005. Global Survey of Regulatory Approaches for Power Quality and Reliability. Tech. rep. 1008589. EPRI, Palo Alto, CA, USA, (Mar. 2005), 146. https://www.epri.com/research/products/1008589.

[96] Tilmann Gneiting and Adrian E Raftery. 2007. Strictly Proper Scoring Rules, Prediction, and Estimation. en. *Journal of the American Statistical Association*, 102, 477, (Mar. 2007), 359–378. doi:10.1198/016214506000001437.

[97] Marc Goerigk and Jannis Kurtz. 2023. Data-driven robust optimization using deep neural networks. *Computers & Operations Research*, 151, (Mar. 2023), 106087. doi:10.1016/j.cor.2022.106087.

[98] Vineet Goyal and Julien Grand-Clement. 2022. Robust markov decision processes: beyond rectangularity. *Mathematics of Operations Research*.

[99] Christine Gregory, Ken Darby-Dowman, and Gautam Mitra. 2011. Robust optimization and portfolio selection: The cost of robustness. *European Journal of Operational Research*, 212, 2, (July 2011), 417–428. doi:10.1016/j.ejor.2011.02.015.

[100] Weiran Guo, Guanjun Liu, Ziyuan Zhou, Ling Wang, and Jiacun Wang. 2024. Enhancing the robustness of qmix against state-adversarial attacks. *Neurocomputing*, 572, (Mar. 2024), 127191. doi:10.1016/j.neucom.2023.127191.

[101] Yifei Guo, Qiuwei Wu, Houlei Gao, Xinyu Chen, Jacob Østergaard, and Huanhai Xin. 2019. MPC-based coordinated voltage regulation for distribution networks with distributed generation and energy storage system. *IEEE Trans. Sustain. Energy*, 10, 4, (Oct. 2019), 1731–1739. doi:10.1109/TSTE.2018.2869932.

[102] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. (Aug. 2018). doi:10.48550/arXiv.1801.01290.

[103] Hassan Haes Alhelou, Mohamad Esmail Hamedani-Golshan, Takawira Cuthbert Njenda, and Pierluigi Siano. 2019. A survey on power system blackout and cascading events: research motivations and challenges. *Energies*, 12, 4, 682. doi:10.3390/en12040682.

[104] Songyang Han, Sanbao Su, Sihong He, Shuo Han, Haizhao Yang, and Fei Miao. 2022. What is the solution for state-adversarial multi-agent reinforcement learning? *arXiv preprint arXiv:2212.02705*.

[105] Matthew J. Hausknecht and Peter Stone. 2015. Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527. arXiv: 1507.06527.

[106] Kevin He, David Adam, Sarah Han-Oh, and Anqi Liu. 2025. Training-Aware Risk Control for Intensity Modulated Radiation Therapies Quality Assurance with Conformal Prediction. arXiv:2501.08963 [cs]. (Jan. 2025). doi:10.48550/arXiv.2501.08963.

[107] Keyang He, Prashant Doshi, and Bikramjit Banerjee. 2022. Reinforcement learning in many-agent settings under partial observability. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence* (Proceedings of Machine Learning Research). James Cussens and Kun Zhang, (Eds.) Vol. 180. PMLR, (Aug. 2022), 780–789.

[108] Sihong He, Songyang Han, Sanbao Su, Shuo Han, Shaofeng Zou, and Fei Miao. 2023. Robust multi-agent reinforcement learning with state uncertainty. *Transactions on Machine Learning Research*.

[109] Chin Pang Ho, Marek Petrik, and Wolfram Wiesemann. 2018. Fast bellman updates for robust mdps. In *International Conference on Machine Learning*. PMLR, 1979–1988.

[110] Chin Pang Ho, Marek Petrik, and Wolfram Wiesemann. 2021. Partial policy iteration for l1-robust markov decision processes. *Journal of Machine Learning Research*, 22, 275, 1–46.

[111] Dimitar Ho, Hoang Le, John Doyle, and Yisong Yue. 2021. Online robust control of nonlinear systems with large uncertainty. In *Proc. 24th Int. Conf. Artif. Intell. Stat. (AISTATS)*. Vol. 130. PMLR, San Diego, CA, USA, 3475–3483.

[112] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 6840–6851. Retrieved Aug. 20, 2025 from https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

[113] Matthew D Hoffman and Matthew J Johnson. 2016. ELBO surgery: yet another way to carve up the variational evidence lower bound. en. In *NIPS 2016 workshop*.

[114] L. Jeff Hong. 2009. Estimating Quantile Sensitivities. *Operations Research*, 57, 1, (Feb. 2009), 118–130. Publisher: INFORMS. doi:10.1287/opre.1080.0531.

[115] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*. Francis Bach and David Blei, (Eds.) Vol. 37. Series Title: Proceedings of Machine Learning Research. PMLR, Lille, France, (July 2015), 448–456. Retrieved July 12, 2017 from http://arxiv.org/abs/1502.03167.

[116] Shariq Iqbal, Christian A. Schröder de Witt, Bei Peng, Wendelin Boehmer, Shimon Whiteson, and Fei Sha. 2021. Randomized entity-wise factorization for multi-agent reinforcement learning. In *ICML*.

[117] Garud N Iyengar. 2005. Robust dynamic programming. *Mathematics of Operations Research*, 30, 2, 257–280.

[118] Aled James and Daniel Schien. 2019. A low carbon kubernetes scheduler. English. In *6th International Conference on ICT for Sustainability, ICT4S 2019* (CEUR Workshop Proceedings). Vol. 2382. CEUR-WS, (June 2019).

[119] Haeun Jeon, Hyunglip Bae, Minsu Park, Chanyeong Kim, and Woo Chang Kim. 2025. Locally Convex Global Loss Network for Decision-Focused Learning. en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39, 25, (Apr. 2025), 26805–26812. doi:10.1609/aaai.v39i25.34884.

[120] Debesh Jha, Pia H. Smedsrud, Michael A. Riegler, Pål Halvorsen, Thomas de Lange, Dag Johansen, and Håvard D. Johansen. 2020. Kvasir-SEG: A Segmented Polyp Dataset. en. In *MultiMedia Modeling*. Yong Man Ro, Wen-Huang Cheng, Junmo Kim, Wei-Ta Chu, Peng Cui, Jung-Woo Choi, Min-Chun Hu, and Wesley De Neve, (Eds.) Springer International Publishing, Cham, 451–462. ISBN: 978-3-030-37734-2. doi:10.1007/978-3-030-37734-2_37.

[121] Jiashuo Jiang, Xiaocheng Li, and Jiawei Zhang. 2025. Online Stochastic Optimization with Wasserstein-Based Nonstationarity. *Management Science*, (Mar. 2025). Publisher: INFORMS. doi:10.1287/mnsc.2020.03850.

[122] Zhengping Jiang, Anqi Liu, and Benjamin Van Durme. 2025. Conformal Linguistic Calibration: Trading-off between Factuality and Specificity. arXiv:2502.19110 [cs]. (Feb. 2025). doi:10.48550/arXiv.2502.19110.

[123] Chancellor Johnstone and Bruce Cox. 2021. Conformal uncertainty sets for robust optimization. en. In *Proceedings of the Tenth Symposium on Conformal and Probabilistic Prediction and Applications*. PMLR, (Sept. 2021), 72–90. Retrieved Aug. 4, 2024 from https://proceedings.mlr.press/v152/johnstone21a.html.

[124] Anatoli Juditsky and Arkadi Nemirovski. 2022. On well-structured convex–concave saddle point problems and variational inequalities with monotone operators. *Optimization Methods and Software*, 37, 5, (Sept. 2022), 1567–1602. _eprint: https://doi.org/10.1080/10556788.2021.1928121. doi:10.1080/10556788.2021.1928121.

[125] Shyam Sundar Kannan, Vishnunandan LN Venkatesh, and Byung-Cheol Min. 2023. Smart-llm: smart multi-agent robot task planning using large language models. *arXiv preprint arXiv:2309.10062*.

[126] Georgios Karatzinis, Christos Korkas, Michalis Terzopoulos, Christos Tsaknakis, Aliki Stefanopoulou, Iakovos Michailidis, and Elias Kosmatopoulos. 2022. Chargym: An EV Charging Station Model for Controller Benchmarking. en. In (IFIP Advances in Information and Communication Technology). Ilias Maglogiannis, Lazaros Iliadis, John Macintyre, and Paulo Cortez, (Eds.) Springer International Publishing, Cham, 241–252. ISBN: 978-3-031-08341-9. doi:10.1007/978-3-031-08341-9_20.

[127] Erim Kardeş, Fernando Ordóñez, and Randolph W Hall. 2011. Discounted robust stochastic games and an application to queueing control. *Operations research*, 59, 2, 365–382.

[128] David L Kaufman and Andrew J Schaefer. 2013. Robust modified policy iteration. *INFORMS Journal on Computing*, 25, 3, 396–410.

[129] Vassilis Kekatos, Georgios B. Giannakis, and Ross Baldick. 2014. Grid topology identification using electricity prices. In *Prof. IEEE Power Eng. Soc. Gen. Meeting*. National Harbor, MD, USA, 1–5. doi:10.1109/PESGM.2014.6939474.

[130] Sujin Kim, Raghu Pasupathy, and Shane G. Henderson. 2015. A Guide to Sample Average Approximation. en. In *Handbook of Simulation Optimization*. Michael C Fu, (Ed.) Springer, New York, NY, 207–243. ISBN: 978-1-4939-1384-8. doi:10.1007/978-1-4939-1384-8_8.

[131] J. King, A. Clifton, and B. Hodge. 2014. Validation of Power Output for the WIND Toolkit. en. Tech. rep. NREL/TP-5D00-61714, 1159354. (Sept. 2014), NREL/TP–5D00–61714, 1159354. doi:10.2172/1159354.

[132] Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*. (Apr. 2014). Retrieved June 10, 2018 from http://arxiv.org/abs/1312.6114.

[133] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*. San Diego, CA, USA, (May 2015). Retrieved July 22, 2019 from http://arxiv.org/abs/1412.6980.

[134] Alexander Kirillov et al. 2023. Segment Anything. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. ISSN: 2380-7504. (Oct. 2023), 3992–4003. doi:10.1109/ICCV51070.2023.00371.

[135] Shayan Kiyani, George Pappas, Aaron Roth, and Hamed Hassani. 2025. Decision Theoretic Foundations for Conformal Prediction: Optimal Uncertainty Quantification for Risk-Averse Agents. arXiv:2502.02561 [cs]. (Feb. 2025). doi:10.48550/arXiv.2502.02561.

[136] Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de-Mello. 2002. The Sample Average Approximation Method for Stochastic Discrete Optimization. en. *SIAM Journal on Optimization*, 12, 2, (Jan. 2002), 479–502. doi:10.1137/S1052623499363220.

[137] Jens Kober, J. Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32, 11, (Sept. 2013), 1238–1274. doi:10.1177/0278364913495721.

[138] Mykel J. Kochenderfer, Christopher Amato, Girish Chowdhary, Jonathan P. How, Hayley J. Davison Reynolds, Jason R. Thornton, Pedro A. Torres-Carrasquillo, N. Kemal Üre, and John Vian. 2015. *Decision Making Under Uncertainty: Theory and Application*. (1st ed.). The MIT Press, (June 2015). ISBN: 978-0-262-02925-4.

[139] Lingkai Kong et al. 2025. Diffusion Models as Constrained Samplers for Optimization with Unknown Constraints. en. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*. PMLR, (Apr. 2025), 4582–4590. Retrieved Sept. 21, 2025 from https://proceedings.mlr.press/v258/kong25b.html.

[140] Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. 2023. Diffusion Models for Black-Box Optimization. en. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, (July 2023), 17842–17857. Retrieved Sept. 21, 2025 from https://proceedings.mlr.press/v202/krishnamoorthy23a.html.

[141] Pavlo Krokhmal, Tanislav Uryasev, and Jonas Palmquist. 2001. Portfolio optimization with conditional value-at-risk objective and constraints. en. *The Journal of Risk*, 4, 2, (Mar. 2001), 43–68. doi:10.21314/JOR.2002.057.

[142] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. 2018. Accurate Uncertainties for Deep Learning Using Calibrated Regression. en. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, (July 2018), 2796–2804. Retrieved Feb. 9, 2023 from https://proceedings.mlr.press/v80/kuleshov18a.html.

[143] Jun-Young Kwak, Rong Yang, Zhengyu Yin, Matthew E. Taylor, and Milind Tambe. 2010. Teamwork and coordination under model uncertainty in dec-pomdps. In *Proceedings of the 3rd AAAI Conference on Interactive Decision Theory and Game Theory* (AAAIWS'10-03). AAAI Press, 30–36.

[144] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc. Retrieved July 18, 2022 from https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html.

[145] Marcus Lapeyrolerie, Melissa S. Chapman, Kari E. A. Norman, and Carl Boettiger. 2021. Deep Reinforcement Learning for Conservation Decisions. (June 2021). doi:10.48550/arXiv.2106.08272.

[146] Jaeho Lee, Sejun Park, and Jinwoo Shin. 2020. Learning Bounds for Risk-sensitive Learning. In *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 13867–13879. Retrieved Feb. 26, 2025 from https://proceedings.neurips.cc/paper/2020/hash/9f60ab2b55468f104055b16df8f69e81-Abstract.html.

[147] Zachary J. Lee, George Lee, Ted Lee, Cheng Jin, Rand Lee, Zhi Low, Daniel Chang, Christine Ortega, and Steven H. Low. 2021. Adaptive Charging Networks: A Framework for Smart Electric Vehicle Charging. *IEEE Transactions on Smart Grid*, 12, 5, (Sept. 2021), 4339–4350. doi:10.1109/TSG.2021.3074437.

[148] Zachary J. Lee, Tongxin Li, and Steven H. Low. 2019. ACN-Data: Analysis and Applications of an Open EV Charging Dataset. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems* (e-Energy '19). Association for Computing Machinery, New York, NY, USA, (June 2019), 139–149. ISBN: 978-1-4503-6671-7. doi:10.1145/3307772.3328313.

[149] Zachary J. Lee, Sunash Sharma, Daniel Johansson, and Steven H. Low. 2021. ACN-Sim: An Open-Source Simulator for Data-Driven Electric Vehicle Charging Research. *IEEE Transactions on Smart Grid*, 12, 6, (Nov. 2021), 5113–5123. doi:10.1109/TSG.2021.3103156.

[150] Joel Z. Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (AAMAS '17). International Foundation for Autonomous Agents and Multiagent Systems, São Paulo, Brazil, 464–473.

[151] Jordan Lekeufack, Anastasios N. Angelopoulos, Andrea Bajcsy, Michael I. Jordan, and Jitendra Malik. 2024. Conformal Decision Theory: Safe Autonomous Decisions from Imperfect Predictions. arXiv:2310.05921. (May 2024). doi:10.48550/arXiv.2310.05921.

[152] Liu Leqi, Audrey Huang, Zachary Lipton, and Kamyar Azizzadenesheli. 2022. Supervised Learning with General Risk Functionals. en. In *Proceedings of the 39th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, (June 2022), 12570–12592. Retrieved Mar. 1, 2023 from https://proceedings.mlr.press/v162/leqi22a.html.

[153] Haoran Li, Yang Weng, Yizheng Liao, Brian Keel, and Kenneth E. Brown. 2019. Robust hidden topology identification in distribution systems. *arXiv:1902.01365*. doi:10.48550/arXiv.1902.01365.

[154] Lihong Li, Wei Chu, John Langford, Taesup Moon, and Xuanhui Wang. 2012. An unbiased offline evaluation of contextual bandit algorithms with generalized linear models. In *Proceedings of the Workshop on On-line Trading of Exploration and Exploitation 2*. JMLR Workshop and Conference Proceedings, 19–36.

[155] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining* (WSDM '11). Association for Computing Machinery, New York, NY, USA, (Feb. 2011), 297–306. ISBN: 978-1-4503-0493-1. doi:10.1145/1935826.1935878.

[156] Na Li, Guannan Qu, and Munther Dahleh. 2014. Real-time decentralized voltage control in distribution networks. In *Proc. 52nd Annu. Allerton Conf. Commun. Control Comput.* Monticello, IL, USA, (Sept. 2014), 582–588. doi:10.1109/ALLERTON.2014.7028508.

[157] Simin Li, Jun Guo, Jingqiao Xiu, Ruixiao Xu, Xin Yu, Jiakai Wang, Aishan Liu, Yaodong Yang, and Xianglong Liu. 2024. Byzantine robust cooperative multi-agent reinforcement learning as a bayesian game. In *The Twelfth International Conference on Learning Representations*.

[158] Tongxin Li, Ruixiao Yang, Guannan Qu, Yiheng Lin, Steven Low, and Adam Wierman. 2022. Equipping Black-Box Policies with Model-Based Advice for Stable Nonlinear Control. (June 2022). doi:10.48550/arXiv.2206.01341.

[159] Zhechao Li, Saeed Jazebi, and Francisco De Leon. 2017. Determination of the optimal switching frequency for distribution system reconfiguration. *IEEE Trans. Power Del.*, 32, 4, (Aug. 2017), 2060–2069.

[160] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. Rllib: abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3053–3062.

[161] Zhirui Liang, Qi Li, Anqi Liu, and Yury Dvorkin. 2025. Prescribing Decision Conservativeness in Two-Stage Power Markets: A Distributionally Robust End-to-End Approach. In *2025 59th Annual Conference on Information Sciences and Systems (CISS)*. ISSN: 2837-178X. (Mar. 2025), 1–6. doi:10.1109/CISS64860.2025.10944764.

[162] Yizheng Liao, Yang Weng, Meng Wu, and Ram Rajagopal. 2015. Distribution grid topology reconstruction: An information theoretic approach. In *Proc. North Am. Power Symp. (NAPS)*. Charlotte, NC, USA, (Oct. 2015), 1–6. ISBN: 978-1-4673-7389-0. doi:10.1109/NAPS.2015.7335248.

[163] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2019. Continuous control with deep reinforcement learning. (July 2019). doi:10.48550/arXiv.1509.02971.

[164] Alexander Lin and Demba E. Ba. 2023. How to Train Your FALCON: Learning Log-Concave Densities with Energy-Based Neural Networks. en. In *Fifth Symposium on Advances in Approximate Bayesian Inference*. (July 2023). Retrieved May 15, 2024 from https://openreview.net/forum?id=UP1r5-t-ov.

[165] Guangyi Liu, Suzan Iloglu, Michael Caldara, Joseph W Durham, and Michael M. Zavlanos. 2025. Distributionally robust multi-agent reinforcement learning for dynamic chute mapping. In *Forty-second International Conference on Machine Learning*.

[166] Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. 2020. Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness. In vol. 33. Curran Associates, Inc., 7498–7512. Retrieved May 6, 2022 from https://proceedings.neurips.cc/paper/2020/hash/543e83748234f7cbab21aa0ade66565f-Abstract.html.

[167] Zhishuai Liu and Pan Xu. 2024. Distributionally robust off-dynamics reinforcement learning: provable efficiency with linear function approximation. *arXiv preprint arXiv:2402.15399*.

[168] Miao Lu, Han Zhong, Tong Zhang, and Jose Blanchet. 2024. Distributionally robust reinforcement learning with interactive data collection: fundamental hardness and near-optimal algorithms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[169] Shitong Luo, Jiaqi Guan, Jianzhu Ma, and Jian Peng. 2021. A 3D Generative Model for Structure-Based Drug Design. en. *Advances in Neural Information Processing Systems*, 34, (Dec. 2021), 6229–6239. Retrieved Aug. 20, 2025 from https://proceedings.neurips.cc/paper/2021/hash/314450613369e0ee72d0da7f6fee773c-Abstract.html?ref=https://githubhelp.com.

[170] Shaocong Ma, Ziyi Chen, Shaofeng Zou, and Yi Zhou. 2023. Decentralized robust v-learning for solving markov games with model uncertainty. *Journal of Machine Learning Research*, 24, 371, 1–40.

[171] Xutao Ma, Chao Ning, and Wenli Du. 2024. Differentiable Distributionally Robust Optimization Layers. en. In *Proceedings of the 41st International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, (July 2024), 33880–33901. Retrieved Aug. 13, 2025 from https://proceedings.mlr.press/v235/ma24j.html.

[172] Avinash N. Madavan, Nathan Dahlin, Subhonmesh Bose, and Lang Tong. 2024. Risk-Based Hosting Capacity Analysis in Distribution Systems. *IEEE Transactions on Power Systems*, 39, 1, (Jan. 2024), 355–365. Conference Name: IEEE Transactions on Power Systems. doi:10.1109/TPWRS.2023.3238846.

[173] Salish Maharjan, Ashwin M. Khambadkone, and Jimmy Chih Hsien Peng. 2021. Robust constrained model predictive voltage control in active distribution networks. *IEEE Trans. Sustain. Energy*, 12, 1, (Jan. 2021), 400–411. doi:10.1109/TSTE.2020.3001115.

[174] Jayanta Mandi, Victor Bucarey, Maxime Mulamba Ke Tchomba, and Tias Guns. 2022. Decision-Focused Learning: Through the Lens of Learning to Rank. en. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, (June 2022), 14935–14947. Retrieved Aug. 7, 2025 from https://proceedings.mlr.press/v162/mandi22a.html.

[175] Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. 2024. Decision-Focused Learning: Foundations, State of the Art, Benchmark and Future Opportunities. en. *Journal of Artificial Intelligence Research*, 80, (Aug. 2024), 1623–1701. doi:10.1613/jair.1.15320.

[176] Daniel J Mankowitz et al. 2019. Robust reinforcement learning for continuous control with model misspecification. *arXiv preprint arXiv:1906.07516*.

[177] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. (June 2016). doi:10.48550/arXiv.1602.01783.

[178] Volodymyr Mnih et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518, 7540, (Feb. 2015), 529–33. doi:10.1038/nature14236.

[179] Peyman Mohajerin Esfahani and Daniel Kuhn. 2018. Data-driven distributionally robust optimization using the Wasserstein metric: performance guarantees and tractable reformulations. en. *Mathematical Programming*, 171, 1-2, (Sept. 2018), 115–166. doi:10.1007/s10107-017-1172-1.

[180] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. 2020. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 21, 132, 1–62. Retrieved Dec. 22, 2023 from http://jmlr.org/papers/v21/19-346.html.

[181] Daniel K Molzahn, Florian Dörfler, Henrik Sandberg, Steven H Low, Sambuddha Chakrabarti, Ross Baldick, and Javad Lavaei. 2017. A survey of distributed optimization and control algorithms for electric power systems. *IEEE Trans. Smart Grid*, 8, 6, 2941–2962. doi:10.1109/TSG.2017.2720471.

[182] 2023. MOSEK optimizer API for python 10.0.46. (2023). https://docs.mosek.com/10.0/pythonapi/index.html.

[183] Zachary Nado et al. 2022. Uncertainty Baselines: Benchmarks for Uncertainty & Robustness in Deep Learning. In arXiv:2106.04015 [cs]. (Jan. 2022). doi:10.48550/arXiv.2106.04015.

[184] Mariola Ndrio, Avinash N. Madavan, and Subhonmesh Bose. 2021. Pricing Conditional Value at Risk-Sensitive Economic Dispatch. en. In *2021 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, Washington, DC, USA, (July 2021), 01–05. ISBN: 978-1-66540-507-2. doi:10.1109/PESGM46819.2021.9637845.

[185] Arkadi Nemirovski and Alexander Shapiro. 2007. Convex Approximations of Chance Constrained Programs. en. *SIAM Journal on Optimization*, 17, 4, (Jan. 2007), 969–996. doi:10.1137/050622328.

[186] Viet Anh Nguyen, Fan Zhang, Shanshan Wang, Jose Blanchet, Erick Delage, and Yinyu Ye. 2024. Robustifying Conditional Portfolio Decisions via Optimal Transport. arXiv:2103.16451 [q-fin]. (Apr. 2024). doi:10.48550/arXiv.2103.16451.

[187] Thanh Nguyen-Tang, Sunil Gupta, A. Tuan Nguyen, and Svetha Venkatesh. 2021. Offline Neural Contextual Bandits: Pessimism, Optimization and Generalization. en. In *Advances in Neural Information Processing Systems*. (Oct. 2021). Retrieved Nov. 30, 2025 from https://openreview.net/forum?id=sPIFuucA3F.

[188] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved Denoising Diffusion Probabilistic Models. en. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, (July 2021), 8162–8171. Retrieved Aug. 11, 2025 from https://proceedings.mlr.press/v139/nichol21a.html.

[189] Arnab Nilim and Laurent El Ghaoui. 2005. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53, 5, 780–798.

[190] Sima Noorani, Orlando Romero, Nicolo Dal Fabbro, Hamed Hassani, and George J. Pappas. 2025. Conformal Risk Minimization with Variance Reduction. arXiv:2411.01696 [cs]. (Feb. 2025). doi:10.48550/arXiv.2411.01696.

[191] Severin Nowak, Yu Christine Chen, and Liwei Wang. 2020. Measurement-based optimal DER dispatch with a recursively estimated sensitivity model. *IEEE Trans. Power Systems*, 35, 6, (Nov. 2020), 4792–4802.

[192] Nilay Noyan, Gábor Rudolf, and Miguel Lejeune. 2022. Distributionally robust optimization under a decision-dependent ambiguity set with applications to machine scheduling and humanitarian logistics. *INFORMS Journal on Computing*, 34, 2, 729–751. eprint: https://doi.org/10.1287/ijoc.2021.1096. doi:10.1287/ijoc.2021.1096.

[193] Frans A. Oliehoek, Matthijs T. J. Spaan, and Nikos Vlassis. 2008. Optimal and approximate q-value functions for decentralized pomdps. *J. Artif. Int. Res.*, 32, 1, (May 2008), 289–353.

[194] Sindhu Padakandla, Prabuchandran K. J, and Shalabh Bhatnagar. 2020. Reinforcement Learning in Non-Stationary Environments. *Applied Intelligence*, 50, 11, (Nov. 2020), 3590–3606. doi:10.1007/s10489-020-01758-5.

[195] Xiang Pan, Minghua Chen, Tianyu Zhao, and Steven H. Low. 2023. DeepOPF: A Feasibility-Optimized Deep Neural Network Approach for AC Optimal Power Flow Problems. *IEEE Systems Journal*, 17, 1, (Mar. 2023), 673–683. doi:10.1109/JSYST.2022.3201041.

[196] Kishan Panaganti and Dileep Kalathil. 2021. Robust reinforcement learning using least squares policy iteration with provable performance guarantees. In *International Conference on Machine Learning (ICML)*, 511–520.

[197] Kishan Panaganti and Dileep Kalathil. 2021. Sample complexity of model-based robust reinforcement learning. In *2021 60th IEEE Conference on Decision and Control (CDC)*, 2240–2245.

[198] Kishan Panaganti, Zaiyan Xu, Dileep Kalathil, and Mohammad Ghavamzadeh. 2022. Robust reinforcement learning using offline data. *Advances in Neural Information Processing Systems (NeurIPS)*.

[199] Kishan Panaganti, Zaiyan Xu, Dileep Kalathil, and Mohammad Ghavamzadeh. 2022. Robust reinforcement learning using offline data. In *Advances in Neural Information Processing Systems*. S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, (Eds.) Vol. 35. Curran Associates, Inc., 32211–32224. https://proceedings.neurips.cc/paper_files/paper/2022/file/d01bda31bbcd780774ff15b534e03c40-Paper-Conference.pdf.

[200] Seiun Park, Deepjyoti Deka, and Michael Chcrtkov. 2018. Exact topology and parameter estimation in distribution grids with minimal observability. In *Proc. 2018 Power Syst. Comput. Conf.* 1–6. doi:10.23919/PSCC.2018.8442881.

[201] Seyed Shahin Parvar and Hamidreza Nazaripouya. 2022. Optimal Operation of Battery Energy Storage Under Uncertainty Using Data-Driven Distributionally Robust Optimization. *Electric Power Systems Research*, 211, (Oct. 2022), 108180. doi:10.1016/j.epsr.2022.108180.

[202] Adam Paszke et al. 2019. PyTorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Number 721. Curran Associates Inc., Red Hook, NY, USA, (Dec. 2019), 8026–8037. Retrieved Aug. 30, 2024 from.

[203] Yash P. Patel, Sahana Rayan, and Ambuj Tewari. 2024. Conformal Contextual Robust Optimization. en. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, (Apr. 2024), 2485–2493. Retrieved May 18, 2024 from https://proceedings.mlr.press/v238/p-patel24a.html.

[204] PJM Interconnection. 2025. Data Miner. (2025). Retrieved Sept. 22, 2025 from https://dataminer2.pjm.com/list.

[205] Bala Kameshwar Poolla, Ashish R. Hota, Saverio Bolognani, Duncan S. Callaway, and Ashish Cherukuri. 2021. Wasserstein Distributionally Robust Look-Ahead Economic Dispatch. *IEEE Transactions on Power Systems*, 36, 3, (May 2021), 2010–2022. Conference Name: IEEE Transactions on Power Systems. doi:10.1109/TPWRS.2020.3034488.

[206] Ilan Price et al. 2025. Probabilistic weather forecasting with machine learning. en. *Nature*, 637, 8044, (Jan. 2025), 84–90. doi:10.1038/s41586-024-08252-9.

[207] The Building Energy Codes Program. [n. d.] Prototype building models. (). https://www.energycodes.gov/prototype-building-models.

[208] Chengrui Qu, Huiwen Jia, and Pengcheng You. 2025. Decision-dependent distributionally robust optimization with application to dynamic pricing. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, 693–698. doi:10.1109/CDC57313.2025.11312514.

[209] Chengrui Qu, Christopher Yeh, Kishan Panaganti, Eric Mazumdar, and Adam Wierman. 2026. Distributionally robust cooperative multi-agent reinforcement learning with value factorization. In *The Fourteenth International Conference on Learning Representations*. (Apr. 2026). https://openreview.net/forum?id=2T3LOpqIOO.

[210] Guannan Qu and Na Li. 2020. Optimal distributed feedback voltage control under limited reactive power. *IEEE Trans. Power Syst.*, 35, 1, (Jan. 2020), 315–331. doi:10.1109/TPWRS.2019.2931685.

[211] Guannan Qu, Yiheng Lin, Adam Wierman, and Na Li. 2020. Scalable Multi-Agent Reinforcement Learning for Networked Systems with Average Reward. In *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2074–2086. Retrieved Sept. 25, 2025 from.

[212] Quandl WIKI dataset. 2025. Nasdaq Data Link. en. (2025). Retrieved Sept. 23, 2025 from https://data.nasdaq.com.

[213] Anna Grazia Quaranta and Alberto Zaffaroni. 2008. Robust optimization of conditional value at risk and portfolio selection. *Journal of Banking & Finance*, 32, 10, (Oct. 2008), 2046–2056. doi:10.1016/j.jbankfin.2007.12.025.

[214] Ana Radovanovic et al. 2022. Carbon-aware computing for datacenters. *IEEE Transactions on Power Systems*, 38, 2, 1270–1280.

[215] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22, 268, 1–8. http://jmlr.org/papers/v22/20-1364.html.

[216] Hamed Rahimian and Sanjay Mehrotra. 2019. Distributionally Robust Optimization: A Review. *arXiv:1908.05659 [cs, math, stat]*, (Aug. 2019). arXiv: 1908.05659. Retrieved Mar. 18, 2021 from http://arxiv.org/abs/1908.05659.

[217] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21, 178, 1–51.

[218] Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning*. The MIT Press, (Nov. 2005). ISBN: 978-0-262-25683-4. doi:10.7551/mitpress/3206.001.0001.

[219] Yann G. Rebours, Daniel S. Kirschen, Marc Trotignon, and Sbastien Rossignol. 2007. A survey of frequency and voltage control ancillary services—Part I: technical features. *IEEE Trans. Power Syst.*, 22, 1, (Feb. 2007), 350–357. doi:10.1109/TPWRS.2006.888963.

[220] R. Tyrrell Rockafellar and Stanislav Uryasev. 2002. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26, 7, (July 2002), 1443–1471. doi:10.1016/S0378-4266(02)00271-6.

[221] R. Tyrrell Rockafellar and Stanislav Uryasev. 2000. Optimization of conditional value-at-risk. en. *The Journal of Risk*, 2, 3, 21–41. doi:10.21314/JOR.2000.038.

[222] Yaniv Romano, Evan Patterson, and Emmanuel Candes. 2019. Conformalized Quantile Regression. In *Advances in Neural Information Processing Systems*. H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, (Eds.) Vol. 32. Curran Associates, Inc. http://papers.neurips.cc/paper/8613-conformalized-quantile-regression.pdf.

[223] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. en. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (Lecture Notes in Computer Science). Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, (Eds.) Springer International Publishing, Cham, 234–241. ISBN: 978-3-319-24574-4. doi:10.1007/978-3-319-24574-4_28.

[224] Aurko Roy, Huan Xu, and Sebastian Pokutta. 2017. Reinforcement learning under model mismatch. In *Advances in Neural Information Processing Systems*, 3043–3052.

[225] Olga Russakovsky et al. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115, 3, (Dec. 2015), 211–252. doi:10.1007/s11263-015-0816-y.

[226] Utsav Sadana, Abhilash Chenreddy, Erick Delage, Alexandre Forel, Emma Frejinger, and Thibaut Vidal. 2024. A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, (Mar. 2024). doi:10.1016/j.ejor.2024.03.020.

[227] Otmane Sakhi, Pierre Alquier, and Nicolas Chopin. 2023. Pac-bayesian offline contextual bandits with guarantees. In *International Conference on Machine Learning*. PMLR, 29777–29799.

[228] Mikayel Samvelyan et al. 2019. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (AAMAS '19). International Foundation for Autonomous Agents and Multiagent Systems, Montreal QC, Canada, 2186–2188. ISBN: 9781450363099.

[229] Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. 2025. A Diffusion Model Framework for Unsupervised Neural Combinatorial Optimization. arXiv:2406.01661 [cs]. (Aug. 2025). doi:10.48550/arXiv.2406.01661.

[230] Philipp Schiele, Eric Sager Luxenberg, and Stephen P. Boyd. 2023. Disciplined Saddle Programming. en. *Transactions on Machine Learning Research*, (June 2023). Retrieved Aug. 26, 2025 from https://openreview.net/forum?id=KhMLfEIoUm.

[231] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. (Aug. 2017). doi:10.48550/arXiv.1707.06347.

[232] Tomonobu Senjyu, Yoshitaka Miyazato, Atsushi Yona, Naomitsu Urasaki, and Toshihisa Funabashi. 2008. Optimal distribution voltage control and coordination with distributed generation. *IEEE Trans. Power Del.*, 23, 2, (Apr. 2008), 1236–1242. doi:10.1109/TPWRD.2007.908816.

[233] Glenn Shafer and Vladimir Vovk. 2008. A Tutorial on Conformal Prediction. *Journal of Machine Learning Research*, 9, 12, 371–421. http://jmlr.org/papers/v9/shafer08a.html.

[234] Sanket Shah, Kai Wang, Bryan Wilder, Andrew Perrault, and Milind Tambe. 2022. Decision-Focused Learning without Decision-Making: Learning Locally Optimized Decision Losses. en. *Advances in Neural Information Processing Systems*, 35, (Dec. 2022), 1320–1332. Retrieved Aug. 7, 2025 from https://proceedings.neurips.cc/paper_files/paper/2022/hash/0904c7edde20d7134a77fc7f9cd86ea2-Abstract-Conference.html.

[235] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. 2009. Stochastic Convex Optimization. en. In *Proceedings of Thirty Third Conference on Learning Theory*. PMLR.

[236] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. 2009. *Lectures on Stochastic Programming: Modeling and Theory*. en. Society for Industrial and Applied Mathematics, (Jan. 2009). ISBN: 978-0-89871-687-0 978-0-89871-875-1. doi:10.1137/1.9780898718751.

[237] Keivan Shariatmadar, Neil Yorke-Smith, Ahmad Osman, Fabio Cuzzolin, Hans Hallez, and David Moens. 2025. Generalized Decision Focused Learning under Imprecise Uncertainty–Theoretical Study. arXiv:2502.17984 [cs]. (Mar. 2025). doi:10.48550/arXiv.2502.17984.

[238] Y. Sharon, A. M. Annaswamy, A. L. Motto, and A. Chakraborty. 2012. Topology identification in distribution network with limited measurements. In *Prof. IEEE PES Innov. Smart Grid Technol. (ISGT)*, 1–6. doi:10.1109/ISGT.2012.6175638.

[239] Siqi Shen, Chennan Ma, Chao Li, Weiquan Liu, Yongquan Fu, Songzhu Mei, Xinwang Liu, and Cheng Wang. 2023. Riskq: risk-sensitive multi-agent reinforcement learning value factorization. *Advances in Neural Information Processing Systems*, 36, 34791–34825.

[240] Siqi Shen, Mengwei Qiu, Jun Liu, Weiquan Liu, Yongquan Fu, Xinwang Liu, and Cheng Wang. 2022. Resq: a residual q function-based approach for multi-agent reinforcement learning value factorization. In *NeurIPS*.

[241] Laixi Shi, Gen Li, Yuting Wei, Yuxin Chen, Matthieu Geist, and Yuejie Chi. 2023. The curious price of distributional robustness in reinforcement learning with a generative model. *Advances in Neural Information Processing Systems*, 36.

[242] Laixi Shi, Eric Mazumdar, Yuejie Chi, and Adam Wierman. 2024. Sample-efficient robust multi-agent reinforcement learning in the face of environmental uncertainty. *arXiv preprint arXiv:2404.18909*.

[243] Yuanyuan Shi, Guannan Qu, Steven Low, Anima Anandkumar, and Adam Wierman. 2022. Stability constrained reinforcement learning for real-time voltage control. In *Proc. Am. Control Conf. (ACC)*, 2715–2721. doi:10.23919/ACC53348.2022.9867476.

[244] Juan Silva, Aymeric Histace, Olivier Romain, Xavier Dray, and Bertrand Granado. 2014. Toward embedded detection of polyps in WCE images for early diagnosis of colorectal cancer. en. *International Journal of Computer Assisted Radiology and Surgery*, 9, 2, (Mar. 2014), 283–293. doi:10.1007/s11548-013-0926-3.

[245] David Silver et al. 2017. Mastering the game of Go without human knowledge. *Nature*, 550, 7676, (Oct. 2017), 354–359. doi:10.1038/nature24270.

[246] Mattia Silvestri, Senne Berden, Gaetano Signorelli, Ali İrfan Mahmutoğulları, Jayanta Mandi, Brandon Amos, Tias Guns, and Michele Lombardi. 2026. Score Function Gradient Estimation to Widen the Applicability of Decision-Focused Learning. en. *Journal of Artificial Intelligence Research*, 85, (Feb. 2026). doi:10.1613/jair.1.19498.

[247] Elena Smirnova, Elvis Dohmatob, and Jérémie Mary. 2019. Distributionally Robust Reinforcement Learning. (June 2019). doi:10.48550/arXiv.1902.08708.

[248] James E. Smith and Robert L. Winkler. 2006. The Optimizer's Curse: Skepticism and Postdecision Surprise in Decision Analysis. *Management Science*, 52, 3, (Mar. 2006), 311–322. Publisher: INFORMS. doi:10.1287/mnsc.1050.0451.

[249] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. en. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, (June 2015), 2256–2265. Retrieved Sept. 18, 2025 from https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[250] Kyunghwan Son, Sungsoo Ahn, Roben Delos Reyes, Jinwoo Shin, and Yung Yi. 2020. Qtran++: improved value transformation for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2006.12010*. eprint: 2006.12010.

[251] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. Qtran: learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*. PMLR, 5887–5896.

[252] Yang Song and Stefano Ermon. 2019. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc. Retrieved Sept. 18, 2025 from https://proceedings.neurips.cc/paper_files/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html.

[253] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*. https://openreview.net/forum?id=PxTIG12RRHS.

[254] David Stutz, Krishnamurthy, Dvijotham, Ali Taylan Cemgil, and Arnaud Doucet. 2022. Learning Optimal Conformal Classifiers. arXiv:2110.09192 [cs, stat]. (May 2022). doi:10.48550/arXiv.2110.09192.

[255] Probability Methods Subcommittee. 1979. Ieee reliability test system. *IEEE Transactions on Power Apparatus and Systems*, PAS-98, 6, 2047–2054. doi:10.1109/TPAS.1979.319398.

[256] Chunlin Sun, Linyu Liu, and Xiaocheng Li. 2023. Predict-then-Calibrate: A New Perspective of Robust Contextual LP. en. In *Advances in Neural Information Processing Systems*. Vol. 36. (Dec. 2023), 17713–17741. Retrieved Nov. 21, 2025 from https://proceedings.neurips.cc/paper_files/paper/2023/hash/397271e11322fae8ba7f827c50ca8d9b-Abstract-Conference.html.

[257] Xianzhuo Sun and Jing Qiu. 2021. Two-stage volt/Var control in active distribution networks with multi-agent deep reinforcement learning method. *IEEE Trans. Smart Grid*, 12, 4, (July 2021), 2903–2912. doi:10.1109/TSG.2021.3052998.

[258] Zhiqing Sun and Yiming Yang. 2023. DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization. en. In *Advances in Neural Information Processing Systems*. (Nov. 2023). Retrieved Sept. 22, 2025 from https://openreview.net/forum?id=JV8Ff0lgVV.

[259] Peter Sunehag et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.

[260] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[261] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual Risk Minimization: Learning from Logged Bandit Feedback. en. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, Florence Italy, (May 2015), 939–941. ISBN: 978-1-4503-3473-0. doi:10.1145/2740908.2742564.

[262] Aviv Tamar, Shie Mannor, and Huan Xu. 2014. Scaling up robust mdps using function approximation. In *International Conference on Machine Learning*, 181–189.

[263] Ou Tang and S. Nurmaya Musa. 2011. Identifying risk issues and research advancements in supply chain risk management. *International Journal of Production Economics*. Leading Edge of Inventory Research 133, 1, (Sept. 2011), 25–34. doi:10.1016/j.ijpe.2010.06.013.

[264] Zhiyuan Tang, David J. Hill, and Tao Liu. 2019. Fast distributed reactive power control for voltage regulation in distribution networks. *IEEE Trans. Power Syst.*, 34, 1, (Jan. 2019), 802–805. doi:10.1109/TPWRD.2018.2868158.

[265] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. 2020. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33, 18583–18599.

[266] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. In *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 24804–24816. Retrieved Sept. 19, 2025 from https://proceedings.neurips.cc/paper_files/paper/2021/hash/cfe8504bda37b575c70ee1a8276f3486-Abstract.html.

[267] Jacopo Teneggi, Matthew Tivnan, Web Stayman, and Jeremias Sulam. 2023. How to Trust Your Diffusion Model: A Convex Optimization Approach to Conformal Risk Control. en. In *Proceedings of the 40th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, (July 2023), 33940–33960. Retrieved May 8, 2025 from https://proceedings.mlr.press/v202/teneggi23a.html.

[268] Aurélie Thiele. 2010. A note on issues of over-conservatism in robust optimization with cost uncertainty. *Optimization*, 59, 7, (Oct. 2010), 1033–1040. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/02331930903395592. doi:10.1080/02331930903395592.

[269] L. Thurner, A. Scheidler, F. Schäfer, J. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun. 2018. Pandapower—An open-source python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Trans. Power Syst.*, 33, 6, (Nov. 2018), 6510–6521. doi:10.1109/TPWRS.2018.2829021.

[270] Muhammad Tirmazi, Adam Barker, Nan Deng, Md Ehtesam Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. 2020. Borg: the Next Generation. In *EuroSys'20*. Heraklion, Crete. https://research.google/pubs/pub49065/.

[271] Stanislav Uryasev and R. Tyrrell Rockafellar. 2001. Conditional Value-at-Risk: Optimization Approach. en. In *Stochastic Optimization: Algorithms and Applications*. Stanislav Uryasev and Panos M. Pardalos, (Eds.) Springer US, Boston, MA, 411–435. ISBN: 978-1-4757-6594-6. doi:10.1007/978-1-4757-6594-6_17.

[272] Arash Vahdat and Jan Kautz. 2020. NVAE: A Deep Hierarchical Variational Autoencoder. In *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 19667–19679. Retrieved Sept. 19, 2025 from https://proceedings.neurips.cc/paper/2020/hash/e3b21256183cf7c2c7a66be163579d37-Abstract.html.

[273] Jose R. Vazquez-Canteli, Sourav Dey, Gregor Henze, and Zoltan Nagy. 2020. CityLearn: Standardizing Research in Multi-Agent Reinforcement Learning for Demand Response and Urban Energy Management. (Dec. 2020). doi:10.48550/arXiv.2012.10504.

[274] Abhishek Verma, Luis Pedrosa, Madhukar R. Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the European Conference on Computer Systems (EuroSys)*. Bordeaux, France. https://research.google/pubs/pub43438/.

[275] Daniel Vial, Sanjay Shakkottai, and R Srikant. 2022. Robust multi-agent bandits over undirected graphs. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6, 3, 1–57.

[276] Oriol Vinyals et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. en. *Nature*, 575, 7782, (Nov. 2019), 350–354. doi:10.1038/s41586-019-1724-z.

[277] Oriol Vinyals et al. 2017. Starcraft ii: a new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

[278] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. 2005. *Algorithmic Learning in a Random World*. en. (1st ed.). Springer-Verlag, New York. ISBN: 978-0-387-00152-4. doi:10.1007/b106715.

[279] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. en. In *International Conference on Learning Representations*. (Sept. 2018). Retrieved Apr. 13, 2022 from https://openreview.net/forum?id=rJ4km2R5t7.

[280] Hao Wang and Baosen Zhang. 2018. Energy Storage Arbitrage in Real-Time Markets via Reinforcement Learning. In *2018 IEEE Power Energy Society General Meeting (PESGM)*. (Aug. 2018), 1–5. doi:10.1109/PESGM.2018.8586321.

[281] Irina Wang, Cole Becker, Bart Van Parys, and Bartolomeo Stellato. 2024. Learning Decision-Focused Uncertainty Sets in Robust Optimization. arXiv:2305.19225 [math]. (July 2024). doi:10.48550/arXiv.2305.19225.

[282] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. Qplex: duplex dueling multi-agent q-learning. In *ICLR*.

[283] Lequn Wang, Akshay Krishnamurthy, and Alex Slivkins. 2024. Oracle-Efficient Pessimism: Offline Policy Optimization In Contextual Bandits. en. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, (Apr. 2024), 766–774. Retrieved Dec. 2, 2025 from https://proceedings.mlr.press/v238/wang24a.html.

[284] Prince Zizhuang Wang, Jinhao Liang, Shuyi Chen, Ferdinando Fioretto, and Shixiang Zhu. 2025. Gen-DFL: Decision-Focused Generative Learning for Robust Decision Making. arXiv:2502.05468 [cs]. (Feb. 2025). doi:10.48550/arXiv.2502.05468.

[285] Shengyi Wang, Jiajun Duan, Di Shi, Chunlei Xu, Haifeng Li, Ruisheng Diao, and Zhiwei Wang. 2020. A data-driven multi-agent autonomous voltage control framework using deep reinforcement learning. *IEEE Trans. Power Syst.*, 35, 6, (Nov. 2020), 4644–4654. doi:10.1109/TPWRS.2020.2990179.

[286] Yu-Xiang Wang, Alekh Agarwal, and Miroslav Dudik. 2017. Optimal and Adaptive Off-policy Evaluation in Contextual Bandits. en. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, (July 2017), 3589–3597. Retrieved Dec. 2, 2025 from https://proceedings.mlr.press/v70/wang17a.html.

[287] Yafei Wang, Bo Pan, Mei Li, Jianya Lu, Lingchen Kong, Bei Jiang, and Linglong Kong. 2024. Sample Average Approximation for Conditional Stochastic Optimization with Dependent Data. en. In *Proceedings of the 41th International Conference on Machine Learning*. (June 2024). Retrieved Aug. 7, 2025 from https://openreview.net/forum?id=YuGnRORkJm.

[288] Yue Wang and Shaofeng Zou. 2022. Policy gradient method for robust reinforcement learning. In *International Conference on Machine Learning*, 23484–23526.

[289] J. Warrington, P. J. Goulart, S. Mariéthoz, and M. Morari. 2012. Robust reserve operation in power systems using affine policies. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. (Dec. 2012), 1111–1117. doi:10.1109/CDC.2012.6425913.

[290] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. 2013. Robust Markov decision processes. *Mathematics of Operations Research*, 38, 1, 153–183.

[291] Bryan Wilder, Bistra Dilkina, and Milind Tambe. 2019. Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization. en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 01, (July 2019), 1658–1665. Number: 01. doi:10.1609/aaai.v33i01.33011658.

[292] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. 2016. Deep Kernel Learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Arthur Gretton and Christian C. Robert, (Eds.) PMLR, Cadiz, Spain, (May 2016), 370–378. Retrieved Jan. 15, 2021 from http://proceedings.mlr.press/v51/wilson16.html.

[293] Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. 2023. Learning Likelihoods with Conditional Normalizing Flows. en. arXiv:1912.00042 [cs]. (Nov. 2023). doi:10.48550/arXiv.1912.00042.

[294] Eric M Wolff, Ufuk Topcu, and Richard M Murray. 2012. Robust control of uncertain markov decision processes with temporal logic specifications. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 3372–3379.

[295] Han Xu and Christopher Yeh. 2025. Robust energy storage operation via generative Wasserstein distributionally robust optimization. In *NeurIPS 2025 Workshop on Tackling Climate Change with Machine Learning*. San Diego, CA, USA, (Dec. 2025). https://www.climatechange.ai/papers/neurips2025/79.

[296] Hanchen Xu, Alejandro D Domínguez-García, Venugopal V Veeravalli, and Peter W Sauer. 2020. Data-driven voltage regulation in radial power distribution systems. *IEEE Trans. Power Syst.*, 35, 3, (May 2020), 2133–2143.

[297] Hanchen Xu, Alejandro D. Domínguez-García, and Peter W. Sauer. 2020. Optimal tap setting of voltage regulation transformers using batch reinforcement learning. *IEEE Trans. Power Syst.*, 35, 3, (May 2020), 1990–2001. doi:10.1109/TPWRS.2019.2948132.

[298] Hanchen Xu, Xiao Li, Xiangyu Zhang, and Junbo Zhang. 2019. Arbitrage of Energy Storage in Electricity Markets with Deep Reinforcement Learning. (May 2019). doi:10.48550/arXiv.1904.12232.

[299] Huan Xu and Shie Mannor. 2012. Distributionally robust Markov decision processes. *Mathematics of Operations Research*, 37, 2, 288–300.

[300] Wenhao Xu, Xuefeng Gao, and Xuedong He. 2023. Regret Bounds for Markov Decision Processes with Recursive Optimized Certainty Equivalents. en. In *Proceedings of the 40th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, (July 2023), 38400–38427. Retrieved May 7, 2025 from https://proceedings.mlr.press/v202/xu23d.html.

[301] Zaiyan Xu, Kishan Panaganti, and Dileep Kalathil. 2023. Improved sample complexity bounds for distributionally robust reinforcement learning. *arXiv preprint arXiv:2303.02783*.

[302] Xiaohe Yan, Chenghong Gu, Xin Zhang, and Furong Li. 2020. Robust Optimization-Based Energy Storage Operation for System Congestion Management. en. *IEEE Systems Journal*, 14, 2, (June 2020), 2694–2702. doi:10.1109/JSYST.2019.2932897.

[303] Wenhao Yang, Liangyu Zhang, and Zhihua Zhang. 2022. Toward theoretical understandings of robust Markov decision processes: sample complexity and asymptotics. *The Annals of Statistics*, 50, 6, 3223–3248.

[304] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. 2020. Qatten: A general framework for cooperative multiagent reinforcement learning. *CoRR*, abs/2002.03939. arXiv: 2002.03939.

[305] Zhouhao Yang, Yihong Guo, Pan Xu, Anqi Liu, and Animashree Anandkumar. 2023. Distributionally Robust Policy Gradient for Offline Contextual Bandits. en. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. PMLR, (Apr. 2023), 6443–6462. Retrieved Dec. 2, 2025 from https://proceedings.mlr.press/v206/yang23f.html.

[306] Christopher Yeh, Nicolas Christianson, Adam Wierman, and Yisong Yue. 2025. Conformal Risk Training: End-to-End Optimization of Conformal Risk Control. In *Advances in Neural Information Processing Systems*. Vol. 38. San Diego, CA, USA, (Dec. 2025), 70056–70092. https://proceedings.neurips.cc/paper_files/paper/2025/hash/6559542f75b4452ebaaf82094c7defb7-Abstract-Conference.html.

[307] Christopher Yeh, Jing Yu, Yuanyuan Shi, and Adam Wierman. 2024. Online learning for robust voltage control under uncertain grid topology. *IEEE Transactions on Smart Grid*, 15, 5, (Sept. 2024), 4754–4764. doi:10.1109/TSG.2024.3383804.

[308] Christopher Yeh, Jing Yu, Yuanyuan Shi, and Adam Wierman. 2022. Robust online voltage control with an unknown grid topology. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*. Association for Computing Machinery, (June 2022), 240–250. ISBN: 9781450393973. doi:10.1145/3538637.3538853.

[309] Christopher Yeh et al. 2021. SustainBench: Benchmarks for Monitoring the Sustainable Development Goals with Machine Learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. (Dec. 2021). doi:10.48550/arXiv.2111.04724.

[310] Christopher Yeh et al. 2023. SustainGym: reinforcement learning environments for sustainable energy systems. In *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., New Orleans, LA, USA, (Dec. 2023), 59464–59476. https://proceedings.neurips.cc/paper_files/paper/2023/hash/ba74855789913e5ed36f87288af79e5b-Abstract-Datasets_and_Benchmarks.html.

[311] Christopher Yeh*, Nicolas Christianson*, Alan Wu, Adam Wierman, and Yisong Yue. 2025. End-to-end conformal calibration for optimization under uncertainty. *Transactions on Machine Learning Research*, (Dec. 2025). https://openreview.net/forum?id=yM8qkT0f9H.

[312] Jiafan Yu, Zhecheng Wang, Arun Majumdar, and Ram Rajagopal. 2018. DeepSolar: A Machine Learning Framework to Efficiently Construct a Solar Deployment Database in the United States. English. *Joule*, 2, 12, (Dec. 2018), 2605–2617. doi:10.1016/j.joule.2018.11.021.

[313] Jiafan Yu, Yang Weng, and Ram Rajagopal. 2018. PaToPa: a data-driven parameter and topology joint estimation framework in distribution grids. *IEEE Trans. Power Syst.*, 33, 4, (July 2018), 4335–4347.

[314] Jing Yu, Varun Gupta, and Adam Wierman. 2023. Online adversarial stabilization of unknown linear time-varying systems. In *Prof. 62nd IEEE Conf. Decis. Control (CDC)*, 8320–8327.

[315] Jing Yu, Dimitar Ho, and Adam Wierman. 2023. Online adversarial stabilization of unknown networked systems. In *Prof. ACM Meas. Anal. Comput. Syst.* Number 1. Vol. 7, 1–43.

[316] Jing Yu, Tianyu Zhang, Omid Ardakanian, and Adam Wierman. 2025. Online Comfort-Constrained HVAC Control via Feature Transfer. In *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems* (E-Energy '25). Association for Computing Machinery, New York, NY, USA, (June 2025), 371–384. ISBN: 979-8-4007-1125-1. doi:10.1145/3679240.3734592.

[317] Lebin Yu, Yunbo Qiu, Quanming Yao, Yuan Shen, Xudong Zhang, and Jian Wang. 2024. Robust communicative multi-agent reinforcement learning with active defense. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence* (AAAI'24/IAAI'24/EAAI'24). AAAI Press. ISBN: 978-1-57735-887-9. doi:10.1609/aaai.v38i16.29708.

[318] Chi Zhang, Yuanyuan Shi, and Yize Chen. 2022. BEAR: Physics-Principled Building Environment for Control and Reinforcement Learning. In *Proceedings of the Fourteenth ACM International Conference on Future Energy Systems* (e-Energy '23). arXiv, (Nov. 2022). doi:10.48550/arXiv.2211.14744.

[319] Huan Zhang, Hongge Chen, Duane S Boning, and Cho-Jui Hsieh. 2021. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations*.

[320] Kaiqing Zhang, TAO SUN, Yunzhe Tao, Sahika Genc, Sunil Mallya, and Tamer Basar. 2020. Robust multi-agent reinforcement learning with model uncertainty. In *Advances in Neural Information Processing Systems*. H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, (Eds.) Vol. 33. Curran Associates, Inc., 10571–10583.

[321] Zhengfei Zhang, Kishan Panaganti, Laixi Shi, Yanan Sui, Adam Wierman, and Yisong Yue. 2024. Distributionally robust constrained reinforcement learning under strong duality. *arXiv preprint arXiv:2406.15788*.

[322] Zhili Zhang, Yanchao Sun, Furong Huang, and Fei Miao. 2023. Safe and robust multi-agent reinforcement learning for connected autonomous vehicles under state perturbations. *arXiv preprint arXiv:2309.11057*.

[323] Zihao Zhao, Christopher Yeh, Lingkai Kong Kong, and Kai Wang. 2026. Diffusion-DFL: decision-focused diffusion models for stochastic optimization. In *The Fourteenth International Conference on Learning Representations*. (Apr. 2026). https://openreview.net/forum?id=uhv3f80jmG.

[324] Qipeng P. Zheng, Jianhui Wang, and Andrew L. Liu. 2015. Stochastic Optimization for Unit Commitment—A Review. *IEEE Transactions on Power Systems*, 30, 4, (July 2015), 1913–1924. Conference Name: IEEE Transactions on Power Systems. doi:10.1109/TPWRS.2014.2355204.

[325] Stephan Zheng, Alexander Trott, Sunil Srinivasa, David C. Parkes, and Richard Socher. 2022. The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science Advances*, 8, 18, (May 2022), eabk2607. doi:10.1126/sciadv.abk2607.

[326] Weifeng Zhong, Kan Xie, Yi Liu, Shengli Xie, and Lihua Xie. 2021. Chance Constrained Scheduling and Pricing for Multi-Service Battery Energy Storage. *IEEE Transactions on Smart Grid*, 12, 6, (Nov. 2021), 5030–5042. Conference Name: IEEE Transactions on Smart Grid. doi:10.1109/TSG.2021.3109140.

[327] Ziyuan Zhou and Guanjun Liu. 2023. Robustness testing for multi-agent reinforcement learning: state perturbations on critical agents. *arXiv preprint arXiv:2306.06136*.

[328] Hao Zhu and Hao Jan Liu. 2016. Fast local voltage control under limited reactive power: optimality and stability analysis. *IEEE Trans. Power Syst.*, 31, 5, (Sept. 2016), 3794–3803. doi:10.1109/TPWRS.2015.2504419.