Building Foundation Agents with Internet Knowledge and Large Language Models

Thesis by
Guanzhi Wang

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

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2026
Defended December 15, 2025

Guanzhi Wang
ORCID: 0009-0004-1921-1796

ACKNOWLEDGEMENTS

Foremost, I extend my deepest gratitude to my advisors, Georgia Gkioxari and Yisong Yue. I consider myself extraordinarily fortunate to have had the opportunity to work with both of you. Thank you for welcoming me into your groups, for placing your trust in me, and for your guidance, patience, kindness, and encouragement throughout this journey. I have always felt that I could turn to you not only as advisors but also as friends whose support shaped both my research and my growth. This dissertation would not have been possible without your support.

To Jim Fan, I am deeply grateful for your mentorship and for shaping my research trajectory. From my time as a master's student at Stanford all the way through my Ph.D., you have played an essential and enduring role—often functioning as a co-advisor. Your insights, support, and unwavering belief in my work have profoundly influenced my research path.

I would like to thank my Ph.D. defense committee chair, Katie Bouman. Your thoughtful feedback and guidance have been invaluable in refining both the ideas and presentation of this dissertation.

To the mentors and faculty members who have supported me during my Ph.D.—Anima Anandkumar, Yuke Zhu, Jan Kautz, Yejin Choi, Chaowei Xiao, Dinesh Jayaraman, Osbert Bastani, Ludwig Schmidt—thank you. Each of you has enriched my academic experience immeasurably. It has been a privilege to learn from you, and I am grateful for every interaction.

I have also been incredibly fortunate to collaborate with many brilliant colleagues, including Yinzhen Xu, Yuqi Xie, Fengyuan Hu, Loïc Magne, Ruijie Zheng, Joshua Belofsky, Yunfan Jiang, Ge Yan, Anas Awadalla, Ajay Mandlekar, Jason Ma, Shyamal Buch, Will Liang, Yuncong Yang, Haoyi Zhu, Yanjun Chen, Andrew Tang, De-An Huang, Zhiding Yu, Jonathan Light, Ziniu Hu, Xiusi Chen, Weiqin Chen, Min Cai, Agrim Gupta, Charles Zhang, Yongqiang Dou, Qi Wang, Zhenjia Xu, Ao Zhang, Scott Reed, Avnish Narayan, Joel Jang, and Joohwan Kim. Working with such a talented group has been one of the greatest joys of my Ph.D. career.

I am equally grateful for the wonderful colleagues and close friends I have met at Caltech and beyond—Fengze Xie, Haowen Zhou, Zihui Wu, Hongkai Zheng, Jiawei Zhao, Kaiyu Yang, Zongyi Li, Yujia Huang, Yiheng Lin, Fuzhao Xue, Max Fu, Zhenfei Yin, Rafal Kocielnik, Albert Li, Sabera Talukder, Alex Farhang, Aadarsh

Sahoo, Ziqi Ma, Hao Liu, Julie Wang, Damiano Marsili, Ilona Demler, Raphi Kang, Pan Xu, Bahareh Tolooshams, Sahin Lale, Zhuoran Qiao, Lu Gan, Yus Song, Xingxing Zuo, Dan Zhang, and many others. Your friendship and support have made this journey meaningful and memorable.

I would also like to thank Laura Flower Kim and Daniel Yoder from ISP for your kindness, patience, and efficiency. You made my experience as an international student remarkably smooth and enjoyable.

My sincere thanks extend as well to the advisors who guided me earlier in my academic career—Chi-Keung Tang and Yu-Wing Tai during my undergraduate years, and Fei-Fei Li, Silvio Savarese, and Juan Carlos Niebles during my master's studies. Your inspiring work and mentorship convinced me to pursue a Ph.D. I would not be writing this page if you had not been there at those pivotal moments in my life.

Most importantly, I could not have completed this journey without the unconditional love and unwavering support of my parents, Shujuan Sun and Weidong Wang. Even without formal scientific training, they took a genuine interest in my work, and it has always been a joy to share both my progress and my challenges with them. Thank you for being the most wonderful listeners and for always standing behind me.

Finally, my deepest gratitude goes to my wife, Ziming Zhu—the greatest gift life has given me. We spent 3.5 years in a long-distance relationship, separated by half the Earth, a period that demanded resilience, patience, and trust. Thank you for supporting me through every high and low, for understanding the pressures of my research even when they pulled me away, and for facing every difficulty alongside me. Your belief in me, especially during the moments when I struggled to believe in myself, has been the anchor that carried me through this journey. I am endlessly grateful for your love, your strength, and your unwavering presence in my life.

ABSTRACT

Autonomous agents that perceive, reason, and interact with the world promise a future where intelligent systems assist with daily activities, support work in homes and factories, and take on labor-intensive or repetitive tasks across diverse environments. Tremendous progress has been made across embodied AI, from self-driving cars and autonomous drones to whole-body locomotion and dexterous manipulation, yet building generalist agents that learn continually, generalize broadly, and operate reliably in open-world settings remains an open challenge. Modern systems face three key limitations: the scarcity and cost of collecting large-scale interactive experience; the difficulty of grounding high-level goals into long-horizon behavior; and the fragmentation of perception, language understanding, and motor control across modular, separately trained components.

This dissertation explores a unified and scalable recipe for building *foundation agents*. The core premise is that internet-scale multimodal data, large language models as the embodied reasoning engine, and unified Vision-Language-Action (VLA) architectures together provide a powerful path toward open-ended embodied intelligence. First, internet-scale multimodal data, including gameplay videos, human activity datasets, online tutorials, and wiki documentation, offers unprecedented breadth, exposing agents to diverse strategies, affordances, and environment configurations that cannot be replicated through controlled robotic data collection alone. Second, large language models provide a flexible cognitive layer for planning, task decomposition, iterative self-improvement, tool use, reward generation, and continual skill acquisition without parameter updates. Third, VLA models unify perception, language grounding, and continuous low-level action, enabling fluid real-time behavior across simulation, gaming environments, and physical robotic platforms, while transferring effectively across task families and embodiments.

By integrating these three components, this dissertation advances the development of embodied agents that can draw upon internet-scale knowledge, reason through language, and translate abstract plans into physical behavior. Together, these ideas aim to move us closer to general, open-ended agents that operate with the reliability, adaptability, and competence required to assist humans across a wide range of real-world tasks.

PUBLISHED CONTENT AND CONTRIBUTIONS

Fan, Linxi, Guanzhi Wang*, Yunfan Jiang*, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar (2022). Minedojo: Building Open-Ended Embodied Agents With Internet-Scale Knowledge. In: *Advances in Neural Information Processing Systems* 35, pp. 18343–18362. URL: https://papers.nips.cc/paper_files/paper/2022/file/74a67268c5cc5910f64938cac4526a90-Paper-Datasets_and_Benchmarks.pdf.

G.W. participated in designing the project, developing the method, curating the dataset, running the experiments, and writing the manuscript.

Magne, Loïc*, Anas Awadalla*, Guanzhi Wang*, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, Yisong Yue, Yejin Choi, Yuke Zhu, and Linxi Fan (2026). NitroGen: An Open Foundation Model for Generalist Gaming Agents. In: *arXiv Preprint arXiv:2601.02427*. URL: https://arxiv.org/abs/2601.02427.

G.W. participated in designing the project, developing the method, running the experiments, and writing the manuscript.

Wang, Guanzhi, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar (2024). Voyager: An Open-Ended Embodied Agent With Large Language Models. In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: https://openreview.net/forum?id=ehfRiF0R3a.

G.W. participated in designing the project, developing the method, running the experiments, and writing the manuscript.

Ma, Yecheng Jason, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, and Anima Anandkumar (2024). Eureka: Human-Level Reward Design via Coding Large Language Models. In: *International Conference on Representation Learning* 2024. Ed. by B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, pp. 26516–26560. URL: https://proceedings.iclr.cc/paper_files/paper/2024/file/70c26937fbf3d4600b69a129031b66ec-Paper-Conference.pdf.

G.W. participated in developing the method and writing the manuscript.

Jiang, Yunfan, Agrim Gupta*, Zichen Zhang*, Guanzhi Wang*, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan (July 2023). VIMA: Robot Manipulation With Multimodal Prompts. In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 14975–15022. URL: https://proceedings.mlr.press/v202/jiang23b.html.

G.W. participated in designing the project, developing the method, running the experiments, and writing the manuscript.

NVIDIA, Guanzhi Wang, et al. (2025). Gr00t N1: An Open Foundation Model for Generalist Humanoid Robots. In: *arXiv Preprint arXiv:2503.14734*. URL: https://arxiv.org/abs/2503.14734.
G.W. participated in developing the method, running the experiments, and writing the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Background and Motivation

Autonomous agents that perceive, reason, and interact with the world promise a future where intelligent systems can help with daily activities, assist in homes and factories, and take on labor-intensive or repetitive tasks across a wide range of environments. Tremendous progress has been made across embodied AI, from self-driving cars (Yurtsever et al., 2020; Tesla, Inc., 2025) and autonomous drones (Floreano et al., 2015; Shi et al., 2019; O'Connell et al., 2022) to robotic agents capable of whole-body locomotion (Luo et al., 2025; Cheng et al., 2024) and dexterous manipulation (Rajeswaran et al., 2018; Black et al., 2024). Yet despite these advances, building generalist agents that learn continually, generalize broadly, and operate reliably in open-world environments remains an open challenge.

Developing such general agents requires jointly overcoming limitations in data, embodied reasoning, and model architecture. First, embodied learning is inherently constrained by the difficulty of collecting large-scale interactive experience. Real-world data is expensive and slow to gather (Zhu et al., 2020a; Zhao et al., 2020). Robotic platforms incur significant wear during exploration, and manual resets further limit throughput (Gupta et al., 2021; Chatzilygeroudis et al., 2018). Moreover, laboratory environments cannot capture the diversity of tasks and scenes found in unstructured real-world settings (Open X-Embodiment Collaboration et al., 2024). As a result, agents trained solely on limited physical data struggle to acquire the breadth of experience needed for open-ended competence.

Second, even with sufficient data, conventional policies lack the ability to transform high-level goals into long-horizon, multi-step behavior. End-to-end reinforcement learning (RL) pipelines (Schulman et al., 2017; Haarnoja et al., 2018; Fujimoto et al., 2018) operate directly on control signals and often fail to decompose tasks (Nachum et al., 2018), detect or recover from mistakes once trajectories deviate from training distributions (Garcıa et al., 2015), or incorporate human feedback without retraining (Christiano et al., 2017). This gap between semantic intent and motor execution becomes especially severe in open-ended environments where agents must interpret ambiguous instructions (Fu et al., 2019; Nair et al., 2021), anticipate future events

(Brohan et al., 2023a), or recover from mistakes (Huang et al., 2022b).

Third, traditional model architectures typically separate perception, language understanding, and action, which makes it difficult to coordinate these components in the real world. Independent vision encoders (He et al., 2015; Kostrikov et al., 2020), language models (Brown et al., 2020; Radford et al., 2019), and control policies (Chi et al., 2024a) must be tuned jointly to handle diverse, dynamic, and unpredictable conditions. Such modular pipelines struggle under distribution shifts (Dulac-Arnold et al., 2019) and cannot easily scale to new embodiments or task categories (Driess et al., 2023; NVIDIA et al., 2025; Black et al., 2024).

These challenges motivate the need for a more unified and scalable approach to embodied intelligence. In this dissertation, we explore such an approach: a three-part recipe for building **foundation agents**. The core premise is that internet-scale knowledge, language-based embodied reasoning, and unified Vision-Language-Action (VLA) models together provide a powerful path toward general, open-ended intelligence.

The first component of this recipe is the use of **internet-scale multimodal data** as a training substrate. The internet offers unprecedented breadth through gameplay videos (Fan et al., 2022; Magne et al., 2026), human activity datasets (Miech et al., 2019; Kay et al., 2017), online tutorials, and wiki documentation (Reid et al., 2022). These datasets expose agents to diverse strategies, problem-solving patterns, and environment configurations that are impossible to replicate through controlled robotic data collection alone. Gameplay data supplies open-ended tasks and creative action sequences at an enormous scale, while human videos provide high-level cues about affordances, interactions, and how people behave in everyday settings. Together, they form a broad and complementary foundation for learning transferable, general-purpose behaviors.

The second component is the use of **large language models as the reasoning engine**. LLMs encode extensive world knowledge (Brown et al., 2020), excel at planning (Huang et al., 2022a; Wang et al., 2023a), task decomposition (Brohan et al., 2023a), tool use (Schick et al., 2023), and structured output generation (Liang et al., 2023; Chen et al., 2021c). Embedding LLMs in the perception–action loop enables agents to:

- interpret natural language instructions as actionable plans (Huang et al., 2022a),

- iteratively refine their behavior through environment feedback (Huang et al., 2022b),
- generate reward functions, skill code, or task scaffolds (Ma et al., 2024),
- and acquire new abilities without updating model parameters (Wang et al., 2023a).

LLMs thus bridge the gap between semantic understanding and control, providing a cognitive layer that helps agents structure their behavior in complex environments.

The final component is the development of **VLA foundation models** that unify perception, language grounding, and motor control. These models build upon multimodal transformers (Alayrac et al., 2022; Jiang et al., 2023; Driess et al., 2023), aligning visual understanding and language context with continuous low-level actions. Trained across diverse data sources and embodiments, VLA models can execute fluid, real-time behaviors in both simulated environments (Zhu et al., 2020c; Nasiriany et al., 2024; Raad et al., 2024) and physical robotic platforms (Brohan et al., 2022; Brohan et al., 2023b; Bu et al., 2025), while also transferring skills between gaming, simulation, and real-world tasks (Raad et al., 2024; Magne et al., 2026; Reed et al., 2022).

Together, these three components form a cohesive recipe for building foundation agents: systems that draw upon the breadth of internet knowledge, leverage the reasoning capabilities of LLMs, and employ unified VLA architectures to ground abstract plans into physical behavior. By integrating open-ended exploration, scalable skill acquisition, automated reward synthesis, and cross-embodiment control, this dissertation aims to advance embodied agents toward human-level generality. We hope that the ideas presented herein bring us closer to a future in which intelligent agents, both virtual and physical, operate with the reliability, adaptability, and competence required to assist humans across a broad spectrum of real-world tasks.

1.2 Thesis Outline

Building on the proposed recipe for building foundation agents, this dissertation is organized around its three core components. Chapters 2 and 3 present agent learning methods for leveraging internet-scale multimodal data. Chapters 4 and 5 investigate large language models as the reasoning engine. Chapters 6 and 7 develop VLA models that unify perception, language, and action.

Chapter 2 introduces MineDojo, a novel framework for developing generally capable embodied agents with internet-scale knowledge. Autonomous agents have made great strides in specialist domains like Atari games and Go. However, they typically learn *tabula rasa* in isolated environments with limited and manually conceived objectives, thus failing to generalize across a wide spectrum of tasks and capabilities. Inspired by how humans continually learn and adapt in the open world, we advocate a trinity of ingredients for building generalist agents: 1) an environment that supports a multitude of tasks and goals, 2) a large-scale database of multimodal knowledge, and 3) a flexible and scalable agent architecture. We introduce MineDojo, a new framework built on the popular *Minecraft* game that features a simulation suite with thousands of diverse open-ended tasks and an internet-scale knowledge base with Minecraft videos, tutorials, wiki pages, and forum discussions. Using MineDojo's data, we propose a novel agent learning algorithm that leverages large pre-trained video-language models as a learned reward function. Our agent is able to solve a variety of open-ended tasks specified in free-form language without any manually designed dense shaping reward.

Chapter 3 presents NitroGen, a video-action foundation model for generalist gaming agents that is trained on 40,000 hours of gameplay videos across more than 1,000 games. We incorporate three key ingredients: 1) an internet-scale video-action dataset constructed by automatically extracting player actions from publicly available gameplay videos, 2) a multi-game benchmark environment that can measure cross-game generalization, and 3) a unified vision-action model trained with large-scale behavior cloning. NitroGen exhibits strong competence across diverse domains, including combat encounters in 3D action games, high-precision control in 2D platformers, and exploration in procedurally generated worlds. It transfers effectively to unseen games, and outperforms models trained from scratch.

Chapter 4 proposes Voyager, the first LLM-powered embodied lifelong learning agent in Minecraft that continuously explores the world, acquires diverse skills, and makes novel discoveries without human intervention. Voyager consists of three key components: 1) an automatic curriculum that maximizes exploration, 2) an ever-growing skill library of executable code for storing and retrieving complex behaviors, and 3) a new iterative prompting mechanism that incorporates environment feedback, execution errors, and self-verification for program improvement. Voyager interacts with GPT-4 via blackbox queries, which bypasses the need for model parameter fine-tuning. The skills developed by Voyager are temporally extended,

interpretable, and compositional, which compounds the agent's abilities rapidly and alleviates catastrophic forgetting. Empirically, VOYAGER shows strong in-context lifelong learning capability and exhibits exceptional proficiency in playing Minecraft. VOYAGER is able to utilize the learned skill library in a new Minecraft world to solve novel tasks from scratch, while other techniques struggle to generalize.

Chapter 5 discusses EUREKA, which uses LLMs for robot reward design. LLMs have excelled as high-level semantic planners for sequential decision-making tasks. However, harnessing them to learn complex low-level manipulation tasks, such as dexterous pen spinning, remains an open problem. We bridge this fundamental gap and present EUREKA, a human-level reward design algorithm powered by LLMs. EUREKA exploits the remarkable zero-shot generation, code-writing, and in-context improvement capabilities of state-of-the-art LLMs, such as GPT-4, to perform evolutionary optimization over reward code. The resulting rewards can then be used to acquire complex skills via reinforcement learning. Without any task-specific prompting or pre-defined reward templates, EUREKA generates reward functions that outperform expert human-engineered rewards. The generality of EUREKA also enables a new gradient-free in-context learning approach to reinforcement learning from human feedback (RLHF), readily incorporating human inputs to improve the quality and the safety of the generated rewards without model updating. Finally, using EUREKA rewards in a curriculum learning setting, we demonstrate for the first time, a simulated Shadow Hand capable of performing pen spinning tricks, adeptly manipulating a pen in circles at rapid speed.

Chapter 6 introduces VIMA, a novel multimodal prompting formulation that converts diverse robot manipulation tasks into a uniform sequence modeling problem. Prompt-based learning has emerged as a successful paradigm in natural language processing, where a single general-purpose language model can be instructed to perform any task specified by input prompts. Yet task specification in robotics comes in various forms, such as imitating one-shot demonstrations, following language instructions, and reaching visual goals. They are often considered different tasks and tackled by specialized models. This work shows that we can express a wide spectrum of robot manipulation tasks with *multimodal prompts*, interleaving textual and visual tokens. We design a transformer-based generalist robot agent, VIMA, that processes these prompts and outputs motor actions autoregressively. To train and evaluate VIMA, we develop a new simulation benchmark with thousands of procedurally-generated tabletop tasks with multimodal prompts, 600K+ expert trajectories for imitation

learning, and four levels of evaluation protocol for systematic generalization. VIMA achieves strong scalability in both model capacity and data size.

Chapter 7 presents GR00T N1, an open VLA model for generalist humanoid robots. General-purpose robots need a versatile body and an intelligent mind. Recent advancements in humanoid robots have shown great promise as a hardware platform for building generalist autonomy in the human world. A robot foundation model, trained on massive and diverse data sources, is essential for enabling the robots to reason about novel situations, robustly handle real-world variability, and rapidly learn new tasks. To this end, we introduce GR00T N1, an open foundation model for humanoid robots. GR00T N1 is a VLA model with a dual-system architecture. The vision-language module (System 2) interprets the environment through vision and language instructions. The subsequent diffusion transformer module (System 1) generates fluid motor actions in real time. Both modules are tightly coupled and jointly trained end-to-end. We train GR00T N1 with a heterogeneous mixture of real-robot trajectories, human videos, and synthetically generated datasets. We show that our generalist robot model GR00T N1 outperforms the state-of-the-art imitation learning baselines on standard simulation benchmarks across multiple robot embodiments. Furthermore, we deploy our model on the Fourier GR-1 humanoid robot for language-conditioned bimanual manipulation tasks, achieving strong performance with high data efficiency.

Finally, Chapter 8 concludes with a summary of the core ideas presented in this dissertation and discusses several promising avenues for scaling and advancing foundation agents even further.

Chapter 2

INTERNET KNOWLEDGE FOR OPEN-ENDED AGENT LEARNING

This chapter is based on the paper:

Fan, Linxi, Guanzhi Wang*, Yunfan Jiang*, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar (2022). Minedojo: Building Open-Ended Embodied Agents With Internet-Scale Knowledge. In: *Advances in Neural Information Processing Systems* 35, pp. 18343–18362. URL: https://papers.nips.cc/paper_files/paper/2022/file/74a67268c5cc5910f64938cac4526a90-Paper-Datasets_and_Benchmarks.pdf.

2.1 Introduction

Developing autonomous embodied agents that can attain human-level performance across a wide spectrum of tasks has been a long-standing goal for AI research. There has been impressive progress towards this goal, most notably in games (Mnih et al., 2013; OpenAI et al., 2019; Vinyals et al., 2019a) and robotics (Kolve et al., 2017; Savva et al., 2019; Zhu et al., 2020c; Xia et al., 2019; Shen et al., 2020). These embodied agents are typically trained *tabula rasa* in isolated worlds with limited complexity and diversity. Although highly performant, they are specialist models that do not generalize beyond a narrow set of tasks. In contrast, humans inhabit an infinitely rich reality, continuously learn from and adapt to a wide variety of open-ended tasks, and are able to leverage large amount of prior knowledge from their own experiences as well as others.

We argue that **three main pillars** are necessary for generalist embodied agents to emerge. First, the environment in which the agent acts needs to **enable an unlimited variety of open-ended goals** (Standish, 2003; Langdon, 2005; Taylor et al., 2016; Stanley et al., 2017). Natural evolution is able to nurture an ever-expanding tree of diverse life forms thanks to the infinitely varied ecological settings that the Earth supports (Stanley et al., 2017; Wang et al., 2019). This process has not stagnated for billions of years. In contrast, today's agent training algorithms cease to make new progress after convergence in narrow environments (Mnih et al., 2013; Zhu et al., 2020c). Second, a **large-scale database of prior knowledge** is necessary

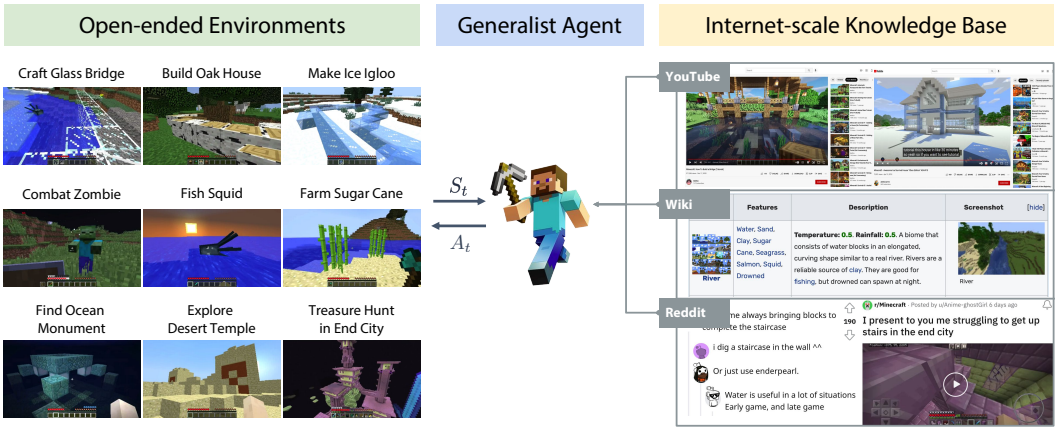


Figure 2.1: MINEDOJO is a novel framework for developing open-ended, generally capable agents that can learn and adapt continually to new goals. MINEDOJO features a benchmarking suite with *thousands* of diverse open-ended tasks specified in natural language prompts, and also provides an internet-scale, multimodal knowledge base of YouTube videos, Wiki pages, and Reddit posts. The database captures the collective experience and wisdom of millions of Minecraft gamers for an AI agent to learn from. Best viewed zoomed in.

to facilitate learning in open-ended settings. Just as humans frequently learn from the internet, agents should also be able to harvest practical knowledge encoded in large amounts of video demos (Goyal et al., 2017; Miech et al., 2019), multimedia tutorials (*Minecraft Wiki* 2016), and forum discussions (Vølske et al., 2017; Kim et al., 2019; Henderson et al., 2019). In a complex world, it would be extremely inefficient for an agent to learn everything from scratch through trial and error. Third, the **agent's architecture** needs to be flexible enough to pursue any task in open-ended environments, and scalable enough to convert large-scale knowledge sources into actionable insights (Chen et al., 2021b; Reid et al., 2022). This motivates the design of an agent that has a unified observation/action space, conditions on natural language task prompts, and adopts the Transformer pre-training paradigm (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020) to internalize knowledge effectively.

In light of these three pillars, we introduce MINEDOJO, a new framework for developing open-ended, generally-capable agents. It is built on the popular Minecraft game, where a player explores a procedurally generated 3D world with diverse types of terrains to roam, materials to mine, tools to craft, structures to build, and wonders to discover. Unlike most other games (Mnih et al., 2013; OpenAI et al., 2019; Vinyals et al., 2019a), Minecraft defines no specific reward to maximize and no fixed storyline to follow, making it well suited for developing open-ended environments for embodied AI research. We make the following three major contributions:

1. Simulation platform with thousands of diverse open-ended tasks. MINEDOJO provides convenient APIs on top of Minecraft that standardize task specification, world settings, and agent's observation/action spaces. We introduce a benchmark suite that consists of thousands of natural language-prompted tasks, making it *two orders of magnitude* larger than prior Minecraft benchmarks like the MineRL Challenge (Guss et al., 2019a; Kanervisto et al., 2022). The suite includes long-horizon, open-ended tasks that cannot be easily evaluated through automated procedures, such as "*build an epic modern house with two floors and a swimming pool*." Inspired by the Inception score (Salimans et al., 2016) and FID score (Heusel et al., 2017) that are commonly used to assess AI-generated image quality, we introduce a novel agent evaluation protocol using a large video-language model pre-trained on Minecraft YouTube videos. This complements human scoring (Shah et al., 2021) that is precise but more expensive. Our learned evaluation metric has good agreement with human judgment in a subset of the full task suite considered in the experiments.

2. Internet-scale multimodal Minecraft knowledge base. Minecraft has more than 100 million active players (Wikipedia contributors, 2022), who have collectively generated an enormous wealth of data. They record tutorial videos, stream live play sessions, compile recipes, and discuss tips and tricks on forums. MINEDOJO features a massive collection of 730K+ YouTube videos with time-aligned transcripts, 6K+ free-form Wiki pages, and 340K+ Reddit posts with multimedia contents (Fig. 2.3). We hope that this enormous knowledge base can help the agent acquire diverse skills, develop complex strategies, discover interesting objectives, and learn actionable representations automatically.

3. Novel algorithm for embodied agents with large-scale pre-training. We develop a new learning algorithm for embodied agents that makes use of the internet-scale domain knowledge we have collected from the web. Using the massive volume of YouTube videos from MINEDOJO, we train a video-text contrastive model in the spirit of CLIP (Radford et al., 2021), which associates natural language subtitles with their time-aligned video segments. We demonstrate that this learned correlation score can be used effectively as an *open-vocabulary, massively multi-task reward function* for RL training. Our agent solves the majority of 12 tasks in our experiment using the learned reward model (Fig. 2.2). It achieves competitive performance to agents trained with meticulously engineered dense-shaping rewards, and in some cases outperforms them, with up to 73% improvement in success rates. For open-ended

tasks that do not have a simple success criterion, our agents also perform well without any special modifications.

In summary, this paper proposes an open-ended task suite, internet-scale domain knowledge, and agent learning with recent advances on large pre-trained models (Bommasani et al., 2021). We have open-sourced MineDojo's simulator, knowledge bases, algorithm implementations, pretrained model checkpoints, and task curation tools at `https://minedojo.org/`. We hope that MineDojo will serve as an effective starter framework for the community to develop new algorithms and advance towards generally capable embodied agents.

2.2 Related work

Open-ended Environments for Decision-making Agents. There are many environments developed with the goal of open-ended agent learning. Prior works include maze-style worlds (Team et al., 2021b; Wang et al., 2019; Juliani et al., 2019), purely text-based game (Küttler et al., 2020), grid worlds (Chevalier-Boisvert et al., 2019; Cao et al., 2020), browser/GUI-based environments (Shi et al., 2017; Toyama et al., 2021), and indoor simulators for robotics (Abramson et al., 2020; Shen et al., 2020; Srivastava et al., 2021; Fan et al., 2021; Shridhar et al., 2020; Savva et al., 2019; Puig et al., 2018). Minecraft offers an exciting alternative for open-ended agent learning. It is a 3D visual world with procedurally generated landscapes and extremely flexible game mechanics that support an enormous variety of activities. Prior methods in open-ended agent learning (Ecoffet et al., 2019; Huizinga et al., 2022; Wang et al., 2020; Kanitscheider et al., 2021; Dennis et al., 2020) do not make use of external knowledge, but our approach leverages internet-scale database to learn open-vocabulary reward models, thanks to Minecraft's abundance of gameplay data online.

Minecraft for AI Research. The Malmo platform (Johnson et al., 2016) is the first comprehensive release of a Gym-style agent API (Brockman et al., 2016) for Minecraft. Based on Malmo, MineRL (Guss et al., 2019a) provides a codebase and human play trajectories for the annual Diamond Challenge at NeurIPS (Guss et al., 2019b; Guss et al., 2021; Kanervisto et al., 2022). MineDojo's simulator builds upon the pioneering work of MineRL, but greatly expands the API and benchmarking task suite. Other Minecraft benchmarks exist with different focuses. For example, CraftAssist (Gray et al., 2019) and IGLU (Kiseleva et al., 2021) study agents with interactive dialogues. BASALT (Shah et al., 2021) applies human evaluation to 4 open-ended tasks. EvoCraft (Grbic et al., 2021) is designed for

structure building, and Crafter (Hafner, 2021) optimizes for fast experimentation. Unlike prior works, MINEDOJO's core mission is to facilitate the development of generally capable embodied agents using internet-scale knowledge.

Internet-scale Multimodal Knowledge Bases. Big dataset such as Common Crawl (*Common Crawl* 2012), the Pile (Gao et al., 2020), LAION (Schuhmann et al., 2021), YouTube-8M (Abu-El-Haija et al., 2016) and HowTo100M (Miech et al., 2019) have been fueling the success of large pre-trained language models (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020) and multimodal models (Sun et al., 2019; Alayrac et al., 2020; Miech et al., 2020; Zhu et al., 2020b; Amrani et al., 2021; Akbari et al., 2021; Xu et al., 2021). While generally useful for learning representations, these datasets are not specifically targeted at embodied agents. To provide agent-centric training data, RoboNet (Dasari et al., 2019) collects video frames from 7 robot platforms, and Ego4D (Grauman et al., 2022) recruits volunteers to record egocentric videos of household activities. In comparison, MINEDOJO's knowledge base is constructed without human curation efforts, much larger in volume, more diverse in data modalities, and comprehensively covers all aspects of the Minecraft environment.

Embodied Agents with Large-scale Pre-training. Inspired by the success in NLP, embodied agent research (Duan et al., 2022; Batra et al., 2020; Ravichandar et al., 2020; Collins et al., 2021) has seen a surge in adoption of the large-scale pre-training paradigm. The recent advances can be roughly divided into 4 categories. 1) **Novel agent architecture**: Decision Transformer (Chen et al., 2021b; Janner et al., 2021; Zheng et al., 2022) applies the powerful self-attention models to sequential decision making. GATO (Reed et al., 2022) and Unified-IO (Lu et al., 2022) learn a single model to solve various decision-making tasks with different control interfaces. VIMA (Jiang et al., 2023) unifies a wide range of robot manipulation tasks with multimodal prompting. 2) **Pre-training for better representations**: R3M (Nair et al., 2022) trains a general-purpose visual encoder for robot perception on Ego4D videos (Grauman et al., 2022). CLIPort (Shridhar et al., 2021) leverages the pre-trained CLIP model (Radford et al., 2021) to enable free-form language instructions for robot manipulation. 3) **Pre-training for better policies**: AlphaStar (Vinyals et al., 2019a) achieves champion-level performance on StarCraft by imitating from numerous human demos. SayCan (Ahn et al., 2022) leverages large language models (LMs) to ground value functions in the physical world. (Li et al., 2022a) and (Reid et al., 2022) directly reuse pre-trained LMs as policy backbone. VPT (Baker et al., 2022) is a concurrent work that learns an inverse dynamics model from human contractors to pseudo-

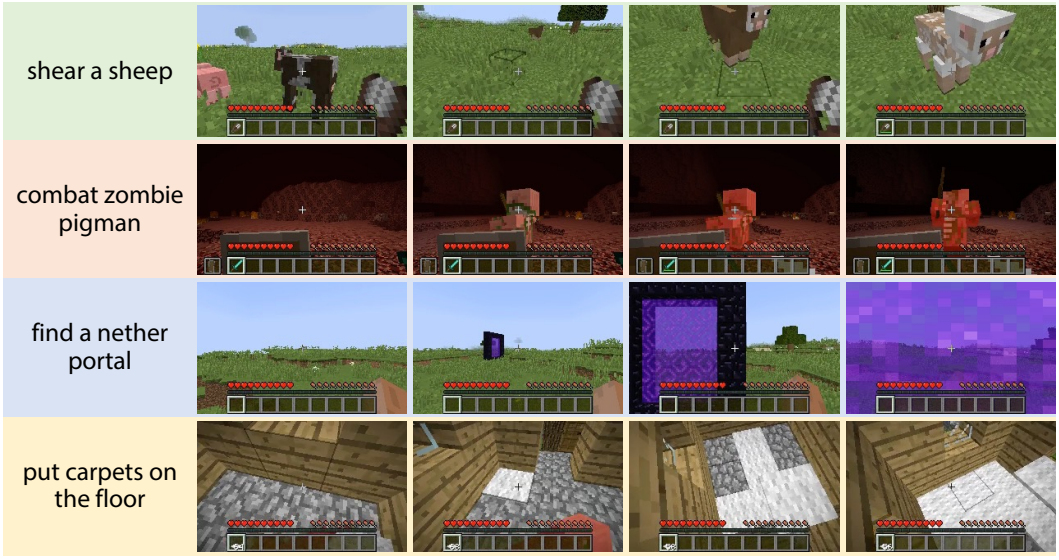


Figure 2.2: Visualization of our agent's learned behaviors on four selected tasks. Leftmost texts are the task prompts used in training. Best viewed on a color display.

label YouTube videos for behavior cloning. VPT is complementary to our approach, and can be finetuned to solve language-conditioned open-ended tasks with our learned reward model. 4) **Data-driven reward functions**: Concept2Robot (Shao et al., 2021) and DVD (Chen et al., 2021a) learn a binary classifier to score behaviors from in-the-wild videos (Goyal et al., 2017). LOReL (Nair et al., 2021) crowd-sources humans labels to train language-conditioned reward function for offline RL. AVID (Smith et al., 2019) and XIRL (Zakka et al., 2021) extract reward signals via cycle consistency. MineDojo's task benchmark and internet knowledge base are generally useful for developing new algorithms in all the above categories. In Sec. 2.5, we also propose an open-vocabulary, multi-task reward model using MineDojo YouTube videos.

2.3 MineDojo Simulator & Benchmark Suite

MineDojo offers a set of simulator APIs help researchers develop generally capable, open-ended agents in Minecraft. It builds upon the open-source MineRL codebase (Guss et al., 2019a) and makes the following upgrades: 1) We provide **unified observation and action spaces** across all tasks, facilitating the development of multi-task and continually learning agents that can constantly adapt to new scenarios and novel tasks. This deviates from the MineRL Challenge design that tailors observation and action spaces to individual tasks; 2) Our simulation unlocks all three types of worlds in Minecraft, including the *Overworld*, the *Nether*, and the *End*, which **substantially expands the possible task space**, while MineRL only

supports the Overworld natively; and 3) We provide convenient APIs to configure initial conditions and world settings to standardize our tasks.

With this MineDojo simulator, we define thousands of benchmarking tasks, which are divided into two categories: 1) *Programmatic tasks* that can be automatically assessed based on the ground-truth simulator states; and 2) *Creative tasks* that do not have well-defined or easily-automated success criteria, which motivates our novel evaluation protocol using a learned model (Sec. 2.5). To scale up the number of Creative tasks, we mine ideas from YouTube tutorials and use OpenAI's GPT-3 (Brown et al., 2020) service to generate substantially more task definitions. Compared to Creative tasks, Programmatic tasks are simpler to get started, but tend to have restricted scope, limited language variations, and less open-endedness in general.

Task Suite I: Programmatic Tasks

We formalize each programmatic task as a 5-tuple: $T = (G, \mathcal{G}, \mathcal{I}, f_{\mathcal{S}}, f_{\mathcal{R}})$. G is an English description of the task goal, such as "*find material and craft a gold pickaxe*." $\mathcal{G}$ is a natural language guidance that provides helpful hints, recipes, or advice to the agent. We leverage OpenAI's `GPT-3-davinci` API to automatically generate detailed guidance for a subset of the tasks. For the example goal "*bring a pig into Nether*," GPT-3 returns: `1) Find a pig in the overworld; 2) Right-click on the pig with a lead; 3) Right-click on the Nether Portal with the lead and pig selected; 4) The pig will be pulled through the portal!` $\mathcal{I}$ is the initial conditions of the agent and the world, such as the initial inventory, spawn terrain, and weather. $f_{\mathcal{S}}$: $s_t \rightarrow \{0, 1\}$ is the success criterion, a deterministic function that maps the current world state s_t to a Boolean success label. $f_{\mathcal{R}}$: $s_t \rightarrow \mathbb{R}$ is an optional dense reward function. We only provide $f_{\mathcal{R}}$ for a small subset of the tasks in MineDojo due to the high costs of meticulously crafting dense rewards. For our current agent implementation (Sec. 2.5), we do not use detailed guidance. Inspired by concurrent works SayCan (Ahn et al., 2022) and Socratic Models (Zeng et al., 2022), one potential idea is to feed each step in the guidance to our learned reward model sequentially so that it becomes a stagewise reward function for a complex multi-stage task.

MineDojo provides 4 categories of programmatic tasks with 1,581 template-generated natural language goals to evaluate the agent's different capabilities systematically and comprehensively:

1. **Survival**: surviving for a designated number of days.
2. **Harvest**: finding, obtaining, cultivating, or manufacturing hundreds of materials and objects.
3. **Tech Tree**: crafting and using a hierarchy of tools.
4. **Combat**: fighting various monsters and creatures that require fast reflex and martial skills.

Each task template has a number of variations based on the terrain, initial inventory, quantity, etc., which form a flexible spectrum of difficulty. In comparison, the NeurIPS MineRL Diamond challenge (Guss et al., 2019a) is a subset of our programmatic task suite, defined by the task goal "*obtain 1 diamond*" in MineDojo.

Task Suite II: Creative Tasks

We define each creative task as a 3-tuple, $T = (G, \mathcal{G}, \mathcal{I})$, which differs from programmatic tasks due to the lack of straightforward success criteria. Inspired by model-based metrics like the Inception score (Salimans et al., 2016) and FID score (Heusel et al., 2017) for image generation, we design a novel task evaluation metric based on a pre-trained contrastive video-language model (Sec. 2.5). In the experiments, we find that the learned metric exhibits a high level of agreement with human evaluations (see Table 2.2).

We brainstorm and author 216 Creative tasks, such as "*build a haunted house with zombie inside*" and "*race by riding a pig*." Nonetheless, such a manual approach is not scalable. Therefore, we develop two systematic approaches to extend the total number of task definitions to 1,560. This makes our Creative tasks *3 orders of magnitude* larger than Minecraft BASALT challenge (Shah et al., 2021), which has 4 Creative tasks.

Approach 1. Task Mining from YouTube Tutorial Videos. We identify our YouTube dataset as a rich source of tasks, as many human players demonstrate and narrate creative missions in the tutorial playlists. To collect high-quality tasks and accompanying videos, we design a multi-stage pipeline that makes it easy to find and annotate interesting tasks. Through this pipeline, we extract 1,042 task ideas from the common wisdom of a huge number of veteran Minecraft gamers, such as "*make an automated mining machine*" and "*grow cactus up to the sky*."

Approach 2. Task Creation by GPT-3. We leverage GPT-3's few-shot capability to generate new task ideas by seeding it with the tasks we manually author or mine from YouTube. The prompt template is: `Here are some example creative tasks in Minecraft: {a few examples}. Let's brainstorm more detailed while reasonable creative tasks in Minecraft`.
GPT-3 contributes 302 creative tasks after de-duplication, and demonstrates a surprisingly proficient understanding of Minecraft terminology.

Collection of Starter Tasks

We curate a set of 64 core tasks for future researchers to get started more easily. If their agent works well on these tasks, they can more confidently scale to the full benchmark.

- **32 programmatic tasks**: 16 "standard" and 16 "difficult," spanning all 4 categories (survival, harvesting, combat, and tech tree). We rely on our Minecraft knowledge to decide the difficulty level. "Standard" tasks require fewer steps and lower resource dependencies to complete.
- **32 creative tasks**: 16 "standard" and 16 "difficult." Similarly, tasks labeled with "standard" are typically short-horizon tasks.

We recommend that researchers run 100 evaluation episodes for each task and report the percentage success rate. The programmatic tasks have ground-truth success, while the creative tasks need our novel evaluation protocol (Sec. 2.6).

2.4 Internet-scale Knowledge Base

Two commonly used approaches (Silver et al., 2017; Vinyals et al., 2019a; OpenAI et al., 2019; Fuchs et al., 2021) to train embodied agents include training agents from scratch using RL with well-tuned reward functions for each task, or using a large amount of human-demonstrations to bootstrap agent learning. However, crafting well-tuned reward functions is challenging or infeasible for our task suite (Sec. 2.3), and employing expert gamers to provide large amounts of demonstration data would also be costly and infeasible (Vinyals et al., 2019a).

Instead, we turn to the open web as an ever-growing, virtually unlimited source of learning material for embodied agents. The internet provides a vast amount of domain knowledge about Minecraft, which we harvest by extensive web scraping and filtering. We collect 33 years worth of YouTube videos, 6K+ Wiki pages,

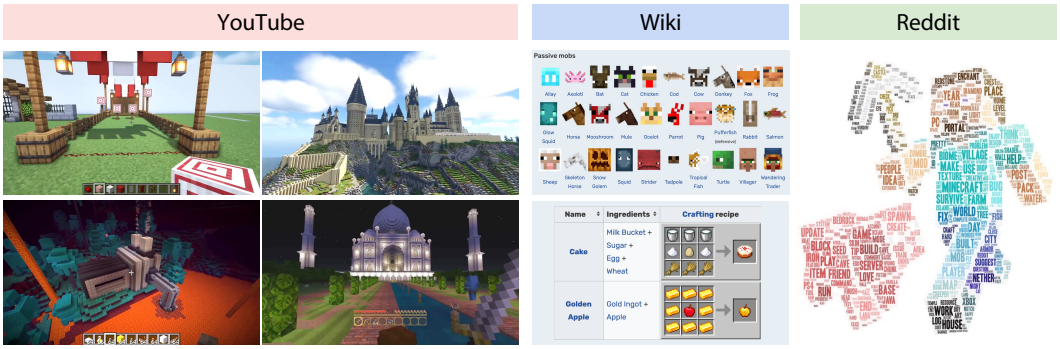


Figure 2.3: MINEDOJO's internet-scale, multimodal knowledge base. Left, YouTube videos: Minecraft gamers showcase the impressive feats they are able to achieve. Clockwise order: an archery range, Hogwarts castle, Taj Mahal, a Nether homebase. Middle, Wiki: Wiki pages contain multimodal knowledge in structured layouts, such as comprehensive catalogs of creatures and recipes for crafting. Right, Reddit: We create a word cloud from Reddit posts and comment threads. Gamers ask questions, share achievements, and discuss strategies extensively. Best viewed zoomed in.

and millions of Reddit comment threads. Instead of hiring a handful of human demonstrators, we capture the collective wisdom of millions of Minecraft gamers around the world. Furthermore, language is a key and pervasive component of our database that takes the form of YouTube transcripts, textual descriptions in Wiki, and Reddit discussions. Language facilitates open-vocabulary understanding, provides grounding for image and video modalities, and unlocks the power of large language models (Devlin et al., 2019; Shoeybi et al., 2019; Brown et al., 2020) for embodied agents. To ensure socially responsible model development, we take special measures to filter out low-quality and toxic contents (Bommasani et al., 2021; Hanu et al., 2020) from our databases.

YouTube Videos and Transcripts. Minecraft is among the most streamed games on YouTube (Gerblick, 2021). Human players have demonstrated a stunning range of creative activities and sophisticated missions that take hours to complete (examples in Fig. 2.3). We collect 730K+ narrated Minecraft videos, which add up to 33 years of duration and 2.2B words in English transcripts. In comparison, HowTo100M (Miech et al., 2019) is a large-scale human instructional video dataset that includes 15 years of experience in total — about half of our volume. The time-aligned transcripts enable the agent to ground free-form natural language in video pixels and learn the semantics of diverse activities without laborious human labeling. We operationalize this insight in our pre-trained video-language model (Sec. 2.5).

Minecraft Wiki. The Wiki pages cover almost every aspect of the game mechanics, and supply a rich source of unstructured knowledge in multimodal tables, recipes, illustrations, and step-by-step tutorials. We use Selenium (*Selenium WebDriver* 2011) to scrape 6,735 pages that interleave text, images, tables, and diagrams. The pages are highly unstructured and do not share any common schema, as the Wiki is meant for human consumption rather than AI training. To preserve the layout information, we additionally save the screenshots of entire pages and extract 2.2M bounding boxes of the visual elements. We do not use Wiki data in our current experiments. Since the Wiki contains detailed recipes for all crafted objects, they could be provided as input or training data for hierarchical planning methods and policy sketches (Andreas et al., 2017). Another promising future direction is to apply document understanding models such as LayoutLM (Xu et al., 2019; Xu et al., 2020) and DocFormer (Appalaraju et al., 2021) to learn actionable knowledge from these unstructured Wiki data.

Reddit. We scrape 340K+ posts along with 6.6M comments under the "r/Minecraft" subreddit. These posts ask questions on how to solve certain tasks, showcase cool architectures and achievements in image/video snippets, and discuss general tips and tricks for players of all expertise levels. We do not use Reddit data for training in Sec. 2.6, but a potential idea is to finetune large language models (Devlin et al., 2019; Radford et al., 2019) on our Reddit corpus to generate instructions and execution plans that are better grounded in the Minecraft domain. Concurrent works (Ahn et al., 2022; Huang et al., 2022b; Zeng et al., 2022) have explored similar ideas and showed excellent results on robot learning, which is encouraging for more future research in MineDojo.

2.5 Agent Learning with Large-scale Pre-training

One of the grand challenges of embodied AI is to build a single agent that can complete a wide range of open-world tasks. The MineDojo framework aims to facilitate new techniques towards this goal by providing an open-ended task suite (Sec. 2.3) and large-scale internet knowledge base (Sec. 2.4). Here we take an initial step towards this goal by developing a proof of concept that demonstrates how a single language-prompted agent can be trained in MineDojo to complete several complex Minecraft tasks. To this end, we propose a novel agent learning algorithm that takes advantage of the massive YouTube data offered by MineDojo. We note that this is only one of the numerous possible ways to use MineDojo's internet database — the Wiki and Reddit corpus also hold great potential to drive

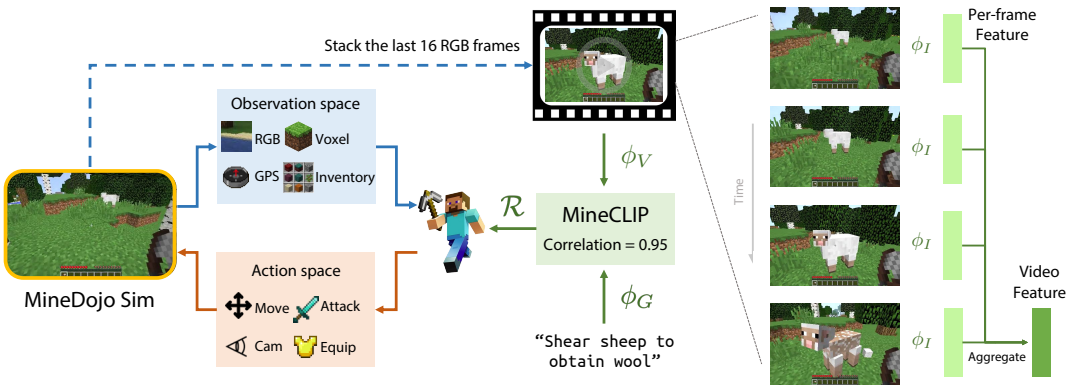


Figure 2.4: Algorithm design. MineCLIP is a contrastive video-language model pre-trained on MineDojo's massive Youtube database. It computes the correlation between an open-vocabulary language goal string and a 16-frame video snippet. The correlation score can be used as a learned dense reward function to train a strong multi-task RL agent.

new algorithm discoveries for the community in future works.

In this paper, we consider a multi-task reinforcement learning (RL) setting, where an agent is tasked with completing a collection of MineDojo tasks specified by language instructions (Sec. 2.3). Solving these tasks often requires the agent to interact with the Minecraft world in a prolonged fashion. Agents developed in popular RL benchmarks (Tassa et al., 2018; Zhu et al., 2020c) often rely on meticulously crafted dense and task-specific reward functions to guide random explorations. However, these rewards are hard or even infeasible to define for our diverse and open-ended tasks in MineDojo. To address this challenge, our key insight is to learn **a dense, language-conditioned reward function from in-the-wild YouTube videos and their transcripts**. Therefore, we introduce **MineCLIP**, a contrastive video-language model that learns to correlate video snippets and natural language descriptions (Fig. 2.4). MineCLIP is multi-task by design, as it is trained on open-vocabulary and diverse English transcripts.

During RL training, MineCLIP provides a high-quality reward signal *without* any domain adaptation techniques, despite the domain gap between noisy YouTube videos and clean simulator-rendered frames. MineCLIP eliminates the need to manually engineer reward functions for each and every MineDojo task. For Creative tasks that lack a simple success criterion (Sec. 2.3), MineCLIP also serves the dual purpose of an **automatic evaluation metric** that agrees well with human judgement on a subset of tasks we investigate (Sec. 2.5, Table 2.2). Because the learned reward model incurs a non-trivial computational overhead, we introduce several techniques

Table 2.1: Our novel MineCLIP reward model is able to achieve competitive performance with manually written dense reward function for Programmatic tasks, and significantly outperforms the $\text{CLIP}_{\text{OpenAI}}$ method across all Creative tasks. Entries represent percentage success rates averaged over 3 seeds, each tested for 200 episodes. Success conditions are precise in Programmatic tasks, but estimated by MineCLIP for Creative tasks.

Group	Tasks	**Ours** (Attn)	**Ours** (Avg)	Manual Reward	Sparse-only	$\text{CLIP}_{\text{OpenAI}}$
	Milk Cow	$\mathbf{64.5 \pm 37.1}$	6.5 ± 3.5	62.8 ± 40.1	0.0 ± 0.0	0.0 ± 0.0
	Hunt Cow	$\mathbf{83.5 \pm 7.1}$	0.0 ± 0.0	48.3 ± 35.9	0.3 ± 0.4	0.0 ± 0.0
	Shear Sheep	12.1 ± 9.1	0.6 ± 0.2	$\mathbf{52.3 \pm 33.2}$	0.0 ± 0.0	0.0 ± 0.0
	Hunt Sheep	8.1 ± 4.1	0.0 ± 0.0	$\mathbf{41.9 \pm 33.0}$	0.3 ± 0.4	0.0 ± 0.0
	Combat Spider	80.5 ± 13.0	60.1 ± 42.5	$\mathbf{87.5 \pm 4.6}$	47.8 ± 33.8	0.0 ± 0.0
	Combat Zombie	47.3 ± 10.6	$\mathbf{72.3 \pm 6.4}$	49.8 ± 26.9	8.8 ± 12.4	0.0 ± 0.0
	Combat Pigman	1.6 ± 2.3	0.0 ± 0.0	$\mathbf{13.6 \pm 9.8}$	0.0 ± 0.0	0.0 ± 0.0
	Combat Enderman	0.0 ± 0.0	0.0 ± 0.0	0.3 ± 0.2	0.0 ± 0.0	0.0 ± 0.0
	Find Nether Portal	37.4 ± 40.8	$\mathbf{89.8 \pm 5.7}$	N/A	N/A	26.3 ± 32.6
	Find Ocean	33.4 ± 45.6	$\mathbf{54.3 \pm 40.7}$	N/A	N/A	9.9 ± 14.1
	Dig Hole	$\mathbf{91.6 \pm 5.9}$	88.1 ± 13.3	N/A	N/A	0.0 ± 0.0
	Lay Carpet	97.6 ± 1.9	$\mathbf{98.8 \pm 1.0}$	N/A	N/A	0.0 ± 0.0

to significantly improve RL training efficiency, making MineCLIP a practical module for open-ended agent learning in Minecraft (Sec. 2.5).

Pre-Training MineCLIP on Large-scale Videos

Formally, the learned reward function can be defined as $\Phi_{\mathcal{R}} : (G, V) \to \mathbb{R}$ that maps a language goal G and a video snippet V to a scalar reward. An ideal $\Phi_{\mathcal{R}}$ should return a high reward if the behavior depicted in the video faithfully follows the language description, and a low reward otherwise. This can be achieved by optimizing the InfoNCE objective (Oord et al., 2018; He et al., 2020; Chen et al., 2020), which learns to correlate positive video and text pairs (Sun et al., 2019; Alayrac et al., 2020; Miech et al., 2020; Akbari et al., 2021; Xu et al., 2021).

Similar to the image-text CLIP model (Radford et al., 2021), MineCLIP consists of a text encoder ϕ_G that embeds a language goal and a video encoder ϕ_V that embeds a moving window of 16 consecutive frames (Fig. 2.4). Our neural architecture has a similar design as CLIP4Clip (Luo et al., 2021), where ϕ_G reuses OpenAI CLIP's pretrained text encoder, and ϕ_V is factorized into a frame-wise image encoder ϕ_I and a temporal aggregator ϕ_a that summarizes the sequence of 16 image features into a single video embedding. Unlike CLIP4Clip, we insert two extra layers of residual CLIP Adapter (Gao et al., 2021) after the aggregator ϕ_a to produce a better video feature, and finetune *only* the last two layers of the pretrained ϕ_I and ϕ_G.

From the MineDojo YouTube database, we follow the procedure in VideoCLIP (Xu et al., 2021) to sample 640K pairs of 16-second video snippets and time-aligned English transcripts, after applying a keyword filter. We train two MineCLIP variants with different types of aggregator ϕ_a: (1) MineCLIP[avg] does simple average pooling, which is fast but agnostic to the temporal ordering; (2) MineCLIP[attn] encodes the sequence by two transformer layers, which is relatively slower but captures more temporal information, and thus produces a better reward signal in general.

RL with MineCLIP Reward

We train a language-conditioned policy network that takes as input raw pixels and predicts discrete control. The policy is trained with PPO (Schulman et al., 2017) on the MineCLIP rewards. In each episode, the agent is prompted with a language goal and takes a sequence of actions to fulfill this goal. When calculating the MineCLIP rewards, we concatenate the agent's latest 16 egocentric RGB frames in a temporal window to form a video snippet. MineCLIP handles all task prompts *zero-shot* without any further finetuning. In our experiments (Sec. 2.6), we show that MineCLIP provides effective dense rewards out of the box, despite the domain shift between in-the-wild YouTube frames and simulator frames. Besides regular video data augmentation, we do not employ any special domain adaptation methods during pre-training. Our finding is consistent with CLIP's strong zero-shot performances on robustness benchmarks in object recognition (Radford et al., 2021).

Compared to hard-coded reward functions in popular benchmarks (Zhu et al., 2020c; Tassa et al., 2018; Fan et al., 2021), the MineCLIP model has 150M parameters and is thus much more expensive to query. We make several design choices to greatly accelerate RL training with MineCLIP in the loop:

1. The language goal G is fixed for a specific task, so the **text features ϕ_G can be precomputed** to avoid invoking the text encoder repeatedly.
2. Our agent's **RGB encoder reuses the pre-trained weights of ϕ_I** from MineCLIP. We do not finetune ϕ_I during RL training, which saves computation and endows the agent with good visual representations from the beginning.
3. MineCLIP's video encoder ϕ_V is factorized into an image encoder ϕ_I and a light-weight aggregator ϕ_a. This design choice enables **efficient image**

Table 2.2: MineCLIP agrees well with the ground-truth human judgment on the Creative tasks we consider. Numbers are F1 scores between MineCLIP's binary classification of tasks success and human labels (scaled to the percentage for better readability).

Tasks	Find Nether Portal	Find Ocean	Dig Hole	Lay Carpet
Ours (Attn)	98.7	**100.0**	99.4	97.4
Ours (Avg)	**100.0**	**100.0**	**100.0**	**98.4**
$\text{CLIP}_{\text{OpenAI}}$	48.7	98.4	80.6	54.1

feature caching. Consider two overlapping video sequences of 8 frames, `V[0:8]` and `V[1:9]`. We can cache the image features of the 7 overlapping frames `V[1]` to `V[7]` to maximize compute savings. If ϕ_V is a monolithic model like S3D (Xie et al., 2018) in VideoCLIP (Xu et al., 2021), then the video features from every sliding window must be recomputed, which would incur a much higher cost per time step.

4. We leverage **Self-Imitation Learning** (Oh et al., 2018) to store the trajectories with high MineCLIP reward values in a buffer, and alternate between PPO and self-imitation gradient steps. It further improves sample efficiency.

2.6 Experiments

We evaluate our agent-learning approach (Section 2.5) on 8 Programmatic tasks and 4 Creative tasks from the MineDojo benchmarking suite. We select these 12 tasks due to the diversity of skills required to solve them (e.g., harvesting, combat, building, navigation) and domain-specific entities (e.g., animals, resources, monsters, terrains, and structures). We split the tasks into 3 groups and train one multi-task agent for each group: `Animal-Zoo` (4 Programmatic tasks on hunting or harvesting resource from animals), `Mob-Combat` (Programmatic, fight 4 types of hostile monsters), and `Creative` (4 tasks).

In the experiments, we empirically check the quality of MineCLIP against manually written reward functions, and quantify how different variants of our learned model affect the RL performance. Table 2.1 presents our main results, and Fig. 2.2 visualizes our learned agent behavior in 4 of the considered tasks. Policy networks of all methods share the same architecture and are trained by PPO + Self-Imitation (Sec. 2.5. We compare the following methods:

Table 2.3: MineCLIP agents have stronger zero-shot visual generalization ability to unseen terrains, weathers, and lighting. Numbers outside parentheses are percentage success rates averaged over 3 seeds (each tested for 200 episodes), while those inside parentheses are relative performance changes.

	Tasks	**Ours** (Attn), train	**Ours** (Attn), unseen test	$\text{CLIP}_{\text{OpenAI}}$, train	$\text{CLIP}_{\text{OpenAI}}$, unseen test
	Milk Cow	64.5 ± 37.1	$\mathbf{64.8 \pm 31.3}(+\ \mathbf{0.8}\%)$	90.0 ± 0.4	$29.2 \pm 3.7\ (-67.6\%)$
	Hunt Cow	83.5 ± 7.1	$\mathbf{55.9 \pm 7.2}\ (\mathbf{-32.9}\%)$	72.7 ± 3.5	$16.7 \pm 1.6\ (-77.0\%)$
	Combat Spider	80.5 ± 13.0	$\mathbf{62.1 \pm 29.7}(\mathbf{-22.9}\%)$	79.5 ± 2.5	$54.2 \pm 9.6\ (-31.8\%)$
	Combat Zombie	47.3 ± 10.6	$\mathbf{39.9 \pm 25.3}(\mathbf{-15.4}\%)$	50.2 ± 7.5	$30.8 \pm 14.4(-38.6\%)$

- **Ours (Attn)**: our agent trained with the MineCLIP[attn] reward model. For Programmatic tasks, we also add the final success condition as a binary reward. For Creative tasks, MineCLIP is the only source of reward.
- **Ours (Avg)**: the average-pooling variant of our method.
- **Manual Reward**: hand-engineered dense reward using ground-truth simulator states.
- **Sparse-only**: the final binary success as a single sparse reward. Note that neither sparse-only nor manual reward is available for Creative tasks.
- $\textbf{CLIP}_{\textbf{OpenAI}}$: pre-trained OpenAI CLIP model that has **not** been finetuned on any MineDojo videos.

MineCLIP is competitive with manual reward. For Programmatic tasks (first 8 rows), RL agents guided by MineCLIP achieve competitive performance as those trained by manual reward. In three of the tasks, they even *outperform* the hand-engineered reward functions, which rely on privileged simulator states unavailable to MineCLIP. For a more statistically sound analysis, we conduct the Paired Student's t-test to compare the mean success rate of each task (pairing column 3 "Ours (Attn)" and column 5 "Manual Reward" in Table 2.1). The test yields p-value $0.3991 \gg 0.05$, which indicates that the difference between our method and manual reward is not considered statistically significant, and therefore they are comparable with each other. Because all tasks require nontrivial exploration, our approach also dominates the Sparse-only baseline. Note that the original OpenAI CLIP model fails to achieve any success. We hypothesize that the creatures in Minecraft look dramatically different from their real-world counterparts, which causes CLIP to produce *misleading* signals worse than no shaping reward at all. It implies the importance of finetuning on MineDojo's YouTube data.

MineCLIP provides automated evaluation. For Creative tasks (last 4 rows), there are no programmatic success criteria available. We convert MineCLIP into a binary success classifier by thresholding the reward value it outputs for an episode. To test the quality of MineCLIP as an automatic evaluation metric, we ask human judges to curate a dataset of 100 successful and 100 failed trajectories for each task. We then run both MineCLIP variants and $\text{CLIP}_{\text{OpenAI}}$ on the dataset and report the binary F1 score of their judgement against human ground-truth in Table 2.2. The results demonstrate that both MineCLIP[attn] and MineCLIP[avg] attain a very high degree of agreement with human evaluation results on this subset of the Creative task suite that we investigate. $\text{CLIP}_{\text{OpenAI}}$ baseline also achieves nontrivial agreement on Find Ocean and Dig Hole tasks, likely because real-world oceans and holes have similar texture. We use the `attn` variant as an automated success criterion to score the 4 Creative task results in Table 2.1. Our proposed method consistently learns better than $\text{CLIP}_{\text{OpenAI}}$-guided agents. It shows that MineCLIP is an effective approach to solving open-ended tasks when no straightforward reward signal is available. We provide further analysis beyond these 4 tasks.

MineCLIP shows good zero-shot generalization to significant visual distribution shift. We evaluate the learned policy without finetuning on a combination of unseen weather, lighting conditions, and terrains — 27 scenarios in total. For the baseline, we train agents with the original $\text{CLIP}_{\text{OpenAI}}$ image encoder (not trained on our YouTube videos) by imitation learning. The robustness against visual shift can be quantitatively measured by the relative performance degradation on novel test scenarios for each task. Table 2.3 shows that while all methods incur performance drops, agents with MineCLIP encoder is more robust to visual changes than the baseline across all tasks. Pre-training on diverse in-the-wild YouTube videos is important to boosting zero-shot visual generalization capability, a finding consistent with literature (Radford et al., 2021; Nair et al., 2022).

Learning a Single Agent for All 12 Tasks We have also trained a single agent for all 12 tasks. To reduce the computational burden without loss of generality, the agent is trained by self-imitating from successful trajectories generated from the self-imitation learning pipeline. We summarize the result in Table 2.4. Similar to our main experiments, all numbers represent percentage success rates averaged over 3 training seeds, each tested for 200 episodes. Compared to the original agents, the new 12-multitask agent sees a performance boost in 6 tasks, degradation in 4

Table 2.4: We train a single multi-task agent for all 12 tasks. All numbers represent percentage success rates averaged over 3 seeds, each tested for 200 episodes.

Group	Tasks	Single Agent on All Tasks	Original	Performance Change
	Milk Cow	**91.5 ± 0.7**	64.5 ± 37.1	$\uparrow$
	Hunt Cow	46.8 ± 3.7	**83.5 ± 7.1**	$\downarrow$
	Shear Sheep	**73.5 ± 0.8**	12.1 ± 9.1	$\uparrow$
	Hunt Sheep	**27.0 ± 1.0**	8.1 ± 4.1	$\uparrow$
	Combat Spider	72.1 ± 1.3	**80.5 ± 13.0**	$\downarrow$
	Combat Zombie	27.1 ± 2.7	**47.3 ± 10.6**	$\downarrow$
	Combat Pigman	**6.5 ± 1.2**	1.6 ± 2.3	$\uparrow$
	Combat Enderman	0.0 ± 0.0	0.0 ± 0.0	$=$
	Find Nether Portal	**99.1 ± 0.4**	37.4 ± 40.8	$\uparrow$
	Find Ocean	**95.1 ± 1.5**	33.4 ± 45.6	$\uparrow$
	Dig Hole	85.8 ± 1.2	**91.6 ± 5.9**	$\downarrow$
	Lay Carpet	96.5 ± 0.8	**97.6 ± 1.9**	$=$

Table 2.5: We test the open-vocabulary generalization ability to two unseen tasks. All numbers represent percentage success rates averaged over 3 seeds, each tested for 200 episodes.

	Tasks	Ours (zero-shot)	Ours (after RL finetune)	Baseline (RL from scratch)
	Hunt Pig	1.3 ± 0.6	**46.0 ± 15.3**	0.0 ± 0.0
	Harvest Spider String	1.6 ± 1.3	**36.5 ± 16.9**	12.5 ± 12.7

tasks, and roughly the same success rates in 2 tasks. This result suggests that there are both positive and negative task transfers happening. To improve the multi-task performance, more advanced algorithms (Yu et al., 2020; Wu et al., 2020) can be employed to mitigate the negative transfer effects.

We also perform Paired Student's *t*-test to statistically compare the performance of the 12-multitask agent and those separately trained on each task group. We obtain a p-value of $0.3720 \gg 0.05$, which suggests that the difference between the two training settings is not statistically significant.

Generalize to Novel Tasks To test the ability to generalize to new open-vocabulary commands, we evaluate the agent on two novel tasks: "harvest spider string" and "hunt pig." Table 2.5 shows that the agent struggles in the zero-shot setting because it has not interacted with pigs or spider strings during training, and thus does not know how to interact with them effectively. However, the performance improves substantially by fine-tuning with the MineCLIP reward. Here the baseline methods are trained from scratch using RL with the MineCLIP encoders and reward. Therefore, the only difference is

whether the policy has been pre-trained on the 12 tasks or not. Given the same environment sampling budget (only around 5% of total samples), it significantly outperforms baselines. It suggests that the multitask agent has learned transferable knowledge on hunting and resource collection, which enables it to quickly adapt to novel tasks.

2.7 Conclusion

In this work, we introduce the MineDojo framework for developing generally capable embodied agents. MineDojo features a benchmarking suite of thousands of Programmatic and Creative tasks, and an internet-scale multimodal knowledge base of videos, wiki, and forum discussions. As an example of the novel research possibilities enabled by MineDojo, we propose MineCLIP as an effective language-conditioned reward function trained with in-the-wild YouTube videos. MineCLIP achieves strong performance empirically and agrees well with human evaluation results, making it a good automatic metric for Creative tasks. We look forward to seeing how MineDojo empowers the community to make progress on the important challenge of open-ended agent learning.

Chapter 3

INTERNET KNOWLEDGE FOR CROSS-EMBODIMENT AGENT LEARNING

Magne, Loïc*, Anas Awadalla*, Guanzhi Wang*, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, Yisong Yue, Yejin Choi, Yuke Zhu, and Linxi Fan (2026). NitroGen: An Open Foundation Model for Generalist Gaming Agents. In: *arXiv Preprint arXiv:2601.02427*. URL: https://arxiv.org/abs/2601.02427.

3.1 Introduction

Building generally capable embodied agents that can operate in unknown environments has long been considered a holy grail of AI research. While computer vision and large language models (LLMs) have achieved this generalization through large-scale pre-training on internet data (Brown et al., 2020; Devlin et al., 2019; Radford et al., 2021; Dosovitskiy et al., 2021), comparable progress in embodied AI has been impeded by the lack of large, diverse, and labeled action datasets. Video games present an ideal domain for advancing embodied AI since they offer visually rich interactive environments and tasks that span a wide range of complexities and temporal horizons. However, prior approaches face substantial limitations. **LLM–based methods** exploit either (1) hand-crafted programmatic APIs to expose internal game states and control agents (Wang et al., 2023a; Volum et al., 2022; Wang et al., 2024b) or (2) complicated perception modules for textual information extraction and object detection (Tan et al., 2024). They enable complex task-solving but require complicated domain-specific design and tuning. **Reinforcement learning** has achieved superhuman performance in individual games such as StarCraft II and Dota 2, but these agents are narrow, costly to train, and depend on specialized simulators rarely available for arbitrary games (Berner et al., 2019; Silver et al., 2016; Vinyals et al., 2019b; Mnih et al., 2013; Mnih et al., 2015). **Behavior-cloning approaches** based on pixel observations have relied on expensive-to-collect demonstrations, constraining training to only a few game titles due to prohibitive data collection costs (Baker et al., 2022; Raad et al., 2024). To date, there has been little progress on developing open-source frameworks that can support the training and evaluation of generalist gaming agents, further hindering progress in this direction.

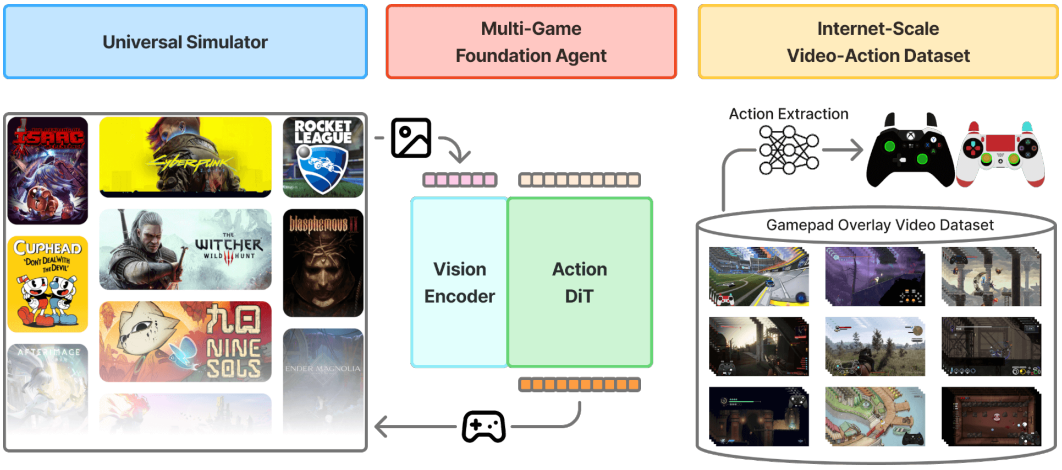


Figure 3.1: NitroGen overview. NitroGen consists of three main components: (1) Multi-game foundation agent (center) — a generalist vision-action model that takes in game observations and generates gamepad actions, enabling zero-shot gameplay across multiple titles and serving as a foundation for fine-tuning on new games; (2) Universal simulator (left) — an environment wrapper that allows any commercial game to be controlled through a Gymnasium API; and (3) Internet-scale dataset (right) — the largest and most diverse open-source gaming dataset curated from 40,000 hours of publicly available gaming videos, spanning more than 1,000 games with extracted action labels.

To address these limitations, we introduce NitroGen, an open foundation model for video game environments trained on 40,000 hours of publicly available internet videos covering more than 1,000 games. We make three major contributions (Figure 3.1):

1. Internet-scale dataset of action-labeled videos. We propose to use a new source of data from publicly available videos where content creators overlay their input commands in real time. We train an annotation model to extract frame-level actions with high accuracy, removing the need for costly manual data collection and capturing a wide spectrum of real player behaviors. Using this approach, we curate a dataset of 40,000 hours of video spanning more than 1,000 games, providing diverse demonstrations for large-scale training.

2. Multi-task multi-game evaluation suite. To assess generalization in realistic settings, we design benchmark environments that comprise 30 tasks of varied complexity from 10 commercial games, covering diverse challenges such as combat, navigation, decision-making, platforming, exploration, and puzzle-solving. This benchmark reflects the demands of modern game environments, where agents must learn to adapt across heterogeneous mechanics and objectives. We provide a universal Gymnasium API (Towers et al., 2024) for our evaluation suite that allows users to

wrap any game to test diverse agent capabilities. This API is what we refer to as the **universal simulator** in Figure 3.1.

3. Large-scale behavior-cloning pre-training. To demonstrate the feasibility and benefits of internet-scale pre-training, we train a vision-action transformer model on our dataset. We demonstrate strong results on our benchmark suite, validating our end-to-end pipeline and showing that it is possible to train a strong multi-game policy using only noisy internet data. We show the benefits of behavior-cloning pre-training by post-training our base model on games not seen during training. The model fine-tuned from the pre-trained NitroGen weights shows up to 52% relative improvement in success rates over the model trained from scratch, given a fixed data and compute budget.

We will open-source the NitroGen dataset, simulator, and pre-trained weights upon acceptance of the paper. We envision NitroGen as a foundational resource that will enable the research community to accelerate progress toward building more generalist embodied agents, fostering new algorithms, model architectures, and applications in this emerging area.

3.2 Related Works

Gaming agents. Video games have long been testbeds for AI, with approaches generally following three directions. Reinforcement learning achieved landmark successes from Atari with DQN (Mnih et al., 2013; Mnih et al., 2015) to AlphaGo (Silver et al., 2016), AlphaStar (Vinyals et al., 2019b), and OpenAI Five (Berner et al., 2019), but these rely on engineered rewards, hand-crafted features, and specialized simulators. More recent vision-based methods like Dreamer 3 (Hafner et al., 2023) still require dedicated simulators and environment-specific training. A second line leverages large language models for high-level reasoning with structured APIs, as in Voyager (Wang et al., 2023a) and Cradle (Tan et al., 2024), but these depend on hand-crafted interfaces. A third category learns directly from pixels or states via behavior cloning, including MineRL (Guss et al., 2019a), VPT (Baker et al., 2022), SIMA (Raad et al., 2024), GATO (Reed et al., 2022), Dreamer 4 Hafner et al., 2025, Lumine Tan et al., 2025, and Farhang et al. Farhang et al., 2024, but they all rely on datasets bootstrapped from human demonstrations or RL-generated data. NitroGen advances this third direction by scaling behavior cloning to internet-scale, enabling training across hundreds of games without costly collection. Game-TARS (Wang et al., 2025) is a concurrent work that also trains a multi-game agent. They combine

contractor data and multi-modal reasoning data to train on a total of 20,000 hours.

Embodied foundation models. Foundation models for embodied AI generally adopt either hierarchical reasoning or end-to-end learning. Hierarchical methods pair pre-trained LLMs or VLMs with low-level policies (Ahn et al., 2022; Driess et al., 2023; Huang et al., 2022b; Liang et al., 2023; Singh et al., 2023) treating the foundation models as black-boxes. Vision-Language-Action (VLA) models (NVIDIA et al., 2025; Kim et al., 2024; Black et al., 2024; Brohan et al., 2022; Cheang et al., 2024; Wen et al., 2025; Team et al., 2024) instead train policies end-to-end on embodied data, though generalizing across tasks and embodiments remains challenging. NitroGen differs by discarding language conditioning and focusing purely on scalable vision-action mapping using diverse gameplay data.

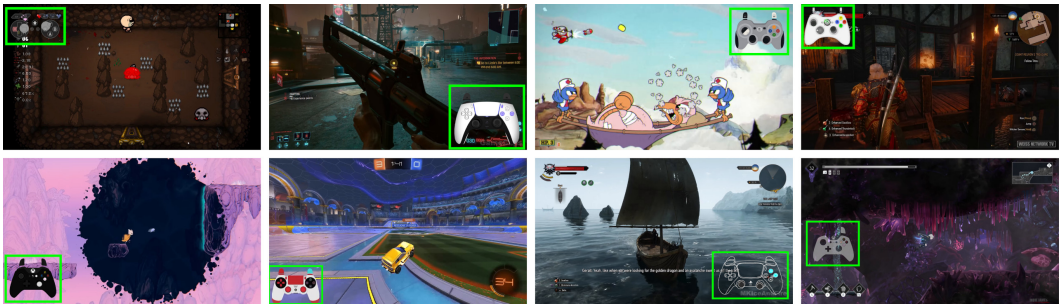
Large-scale action datasets. Progress in vision and NLP has been driven by large labeled datasets, but embodied AI lags behind due to the difficulty of collecting action-labeled data and defining standardized action spaces. Gaming datasets like MineRL (Guss et al., 2019a) provide limited coverage, while MineDojo (Fan et al., 2022) scales video data without action labels. VPT (Baker et al., 2022) annotates 70,000 hours via inverse dynamics but is limited to Minecraft. Other work seeks to infer latent actions from videos (Edwards et al., 2019; Ye et al., 2025; Bruce et al., 2024; Parker-Holder et al., 2024), though scalability is unclear. In robotics, teleoperation has produced datasets such as Roboturk (Mandlekar et al., 2018; Mandlekar et al., 2019; Mandlekar et al., 2020), ALOHA (Aldaco et al., 2024), TeleMoMa (Dass et al., 2024), Open X-Embodiment (O'Neill et al., 2024), and AgiBot World (Bu et al., 2025), but these are costly, limited in scale, and lack organic diversity. NitroGen introduces a scalable alternative by leveraging input overlay software, which naturally provides action labels in publicly available gameplay videos.

3.3 Approach

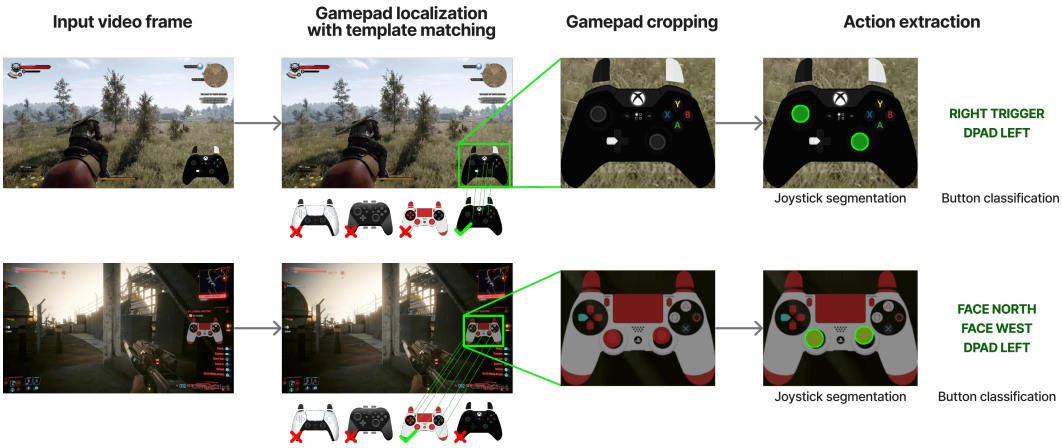
NitroGen consists of three novel components: (1) an internet-scale video dataset with action labels, (2) a multi-game benchmark with a Gymnasium environment wrapper, and (3) a vision–action model pre-trained through large-scale behavior cloning. In this section, we provide details of each component.

Internet-scale multi-game video-action dataset

Annotation challenge. A central challenge in training policies from internet videos is recovering the corresponding actions, since most gameplay recordings typically



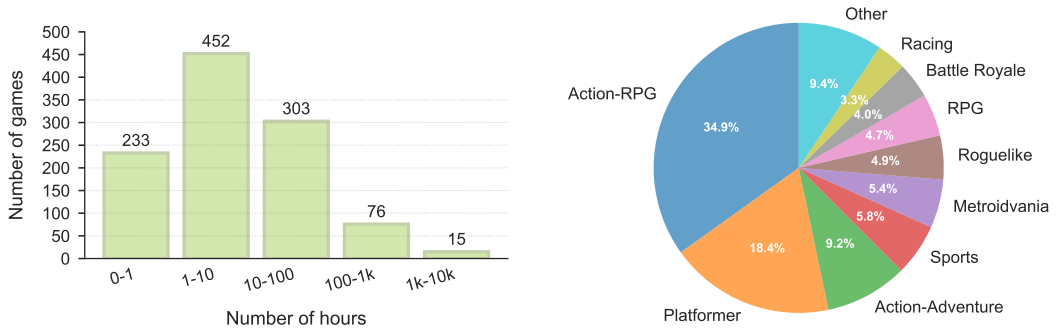
(a) Examples of gamepad overlay videos.



(b) Action extraction pipeline.

Figure 3.2: Video-action dataset pipeline overview. We extract actions from on-screen displays which show the gamepad actions of the player in real-time; called "input overlays." (a) Dataset curation. We collect publicly available videos displaying a "gamepad overlay." The diversity of these overlays presents significant challenges, as gamepads vary widely across content creators in controller types (e.g., Xbox, PlayStation, or others), transparency levels, and visual artifacts introduced by video compression. (b) Action extraction. For each collected video, we localize the gamepad by sampling 25 frames and running keypoint matching against a curated set of templates using SIFT and XFeat features. We use the template-matching results to localize and crop the gamepad region from each video. A hybrid classification–segmentation network is then trained to predict joystick positions and button states from the cropped controller images, enabling accurate reconstruction of player inputs.

do not include the player's inputs. We address this limitation by using a novel source of publicly available videos in which such labels can be recovered. These videos feature *input overlay* software that displays a real-time visualization of the player's actions, typically as a 2D image of a gamepad in a corner of the screen with pressed buttons highlighted (Figure 3.2a).



(a) Distribution of hours per game

(b) Genre distribution by total hours

Figure 3.3: Distribution of the NitroGen dataset across games and genres. After filtering, the NitroGen dataset contains 40,000 hours of gameplay videos spanning more than 1,000 games. (a) Hours per game shows broad coverage, with 846 games having over one hour of data, 91 games with over 100 hours, and 15 games exceeding 1,000 hours each. (b) Genre distribution reveals Action-RPG games are most common (34.9% of total hours), followed by Platformer (18.4%) and Action-Adventure (9.2%) games, with the remainder distributed across seven genres.

Dataset curation. Although input overlays appear in only a fraction of online gameplay videos, they occur frequently enough to enable the construction of a large-scale dataset. We collect 71,000 hours of raw video containing gamepad overlays. While input overlay software was originally used primarily within the speedrunning community, its use has since expanded to many action games and among both expert and casual players. To avoid over-representation of any single title, we use a combination of keyword-based searches and curation guided by content diversity, ensuring coverage across games, genres, and skill levels. This approach balances casual and competitive play styles while maintaining broad genre representation. Figure 3.3 shows the distribution of gameplay hours by title and genre. The dataset covers more than 1,000 unique games, making it the largest labeled video–action dataset for video games to date. It contains 38,739 videos from 818 different content creators, with an average video duration of 1 hour and 50 minutes.

Action extraction. We extract player inputs from gameplay videos through a three-stage pipeline: (1) template matching to locate and crop the gamepad overlay, (2) gamepad action parsing using a fine-tuned segmentation model, and (3) quality filtering to ensure accurate and meaningful data.

Stage 1: Template matching. To locate gamepad overlays within gameplay videos, we apply template matching using a curated set of approximately 300 common controller

templates. For each video, we sample 25 frames and perform feature matching with SIFT (Lowe, 2004) and XFeat (Potje et al., 2024) against all curated templates. We estimate an affine transformation from the paired keypoints and require at least 20 inliers for a match to be considered valid. We then extract the region with the highest matching score, which defines the gamepad location for subsequent processing. Figure 3.2b shows examples of successful match.

Stage 2: Gamepad action parsing. We parse controller states using a fine-tuned SegFormer (Xie et al., 2021) segmentation model that processes pairs of consecutive frames. The model takes two consecutive frames as input (concatenated along the spatial dimension) to capture short-term temporal dynamics. It outputs a segmentation mask to localize joystick positions on a discrete 11×11 grid, and binary button states (Figure 3.2b). Empirically, we find that estimating joystick positions via segmentation masks significantly outperforms direct regression of joystick coordinates.

We train the annotation model using synthetic data generated by sampling frames from the NitroGen training set and programmatically overlaying controller templates using the Open Joystick Display[1], Input Overlay[2], and GamePad Viewer[3] software. For each template, we produce multiple frames with random button states and joystick positions, yielding 8M labeled frames. To simulate real-world visual artifacts, we vary overlay opacity, controller size, and video compression, generating multiple variants per frame. We train the action parsing SegFormer model using the AdamW optimizer (Kingma et al., 2017; Loshchilov et al., 2019) with a learning rate of 0.0001, linear learning rate decay, weight decay of 0.1, and a batch size of 256.

At inference, we compute precise joystick positions by detecting contours for each joystick over the entire video. To estimate the center position of each joystick, we average positions from all frames where the joystick is classified as centered in the 11×11 discrete grid. We then normalize the positions to the range $[-1.0, 1.0]$ using the 99th percentile of absolute x and y values over the video to reduce the influence of outliers.

Stage 3: Quality filtering. The final stage applies targeted filtering strategies to ensure high-quality data. During training, we observe that using the raw 71,000 hours of data leads to the model over-predicting the null action as noted in VPT Baker

[1] `https://github.com/AkikoKumagara/open-joystick-display`
[2] `https://github.com/univrsal/input-overlay`
[3] `https://beta.gamepadviewer.com/`

Figure 3.4: In-game rollouts. We show NitroGen performing tasks in diverse 2D and 3D environments. These tasks can take from a few seconds to a few minutes to perform. Some of them include memorization, while others are performed in procedurally generated worlds and require the model to adapt.

et al., 2022. To avoid that, we discard segments based on action density: we only keep chunks where at least 50% of the timesteps have non-zero button or joystick actions, resulting in 55% of the data being kept. For all gameplay videos, we mask the on-screen controller to prevent models from exploiting it as a shortcut.

Evaluation suite

Universal simulator for any game title. Many research environments provide a Gymnasium API (Towers et al., 2024) that enables programmatic control of the simulation. To bring this capability to commercial video games, which typically lack such an interface, we develop a universal simulator that can wrap any game title with a Gymnasium API for model development. The library intercepts the game engine's system clock to control simulation time, enabling frame-by-frame interaction without modifying game code. This approach works with any title that uses the system clock for physics and interactions, which is a common practice in

game development. We leave real-time or asynchronous deployment to future work. Frequent pausing and resuming during gameplay could potentially affect the game's physics engine in unknown ways, we verify that this process does not alter games' physics and behaviors.

Unified observation and action space. Using this simulator, we introduce a multi-game, multi-task benchmark with a shared interface across all titles. Observations are single RGB frames. Actions consist of a standardized 16-dimensional binary vector for gamepad buttons (4 d-pad buttons, 4 face buttons, 2 shoulders, 2 triggers, 2 joystick thumb buttons, start, back) plus a 4-dimensional continuous vector for joystick positions. Unlike prior work that defines game or task-specific action spaces (Baker et al., 2022; Guss et al., 2019a), this unified layout facilitates direct policy transfer across diverse games.

Diverse evaluation tasks. The evaluation suite serves as a universal evaluation framework for multi-game policies, covering 10 games across diverse visual styles and genres with 30 tasks total. The suite includes five 2D games and five 3D games, each testing different skill combinations. The 2D games include three side-scrollers and two top-down roguelikes with procedurally generated levels. The 3D games consist of two open-world games, two combat-focused action-RPGs, and one sports game.

Tasks are distributed across three categories: 11 combat tasks (boss fights, enemy encounters), 10 navigation tasks (reaching specific locations, traversing environments), and 9 game-specific tasks (unique mechanics particular to individual games). Each task has clearly defined start and goal states, with attempts typically lasting a few minutes, though human players may require several hours of repeated attempts to succeed. We select tasks where the initial visual state provides sufficient context to elicit correct behavior, leaving language-conditioned specifications to future work. Success rates are measured through human evaluation.

NitroGen foundation model

Architecture. Building on recent advances in generative modeling and robotics, NitroGen employs flow matching (Lipman et al., 2022) to generate chunks of future actions conditioned on visual observations. The architecture is adapted from GR00T N1 (NVIDIA et al., 2025) with the language and state encoders removed, and a single action head. RGB inputs at 256×256 resolution are encoded using a SigLIP 2 vision transformer (Tschannen et al., 2025), producing 256 image tokens per frame. Actions

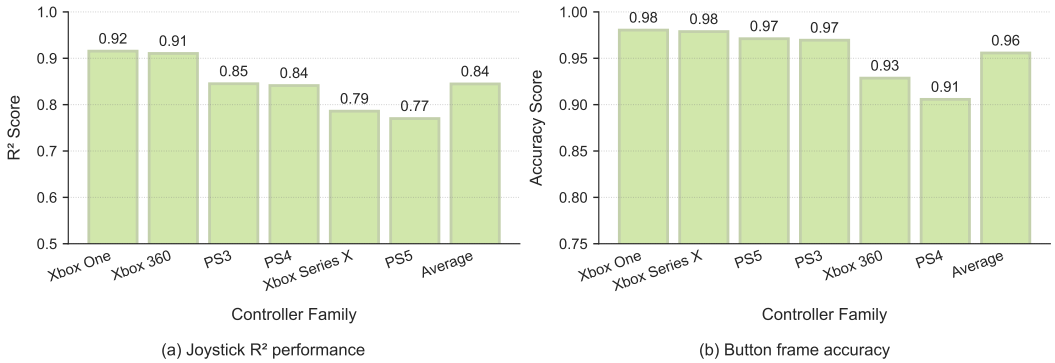


Figure 3.5: Gamepad parsing performance for different controller families. We verify the correctness of our action extraction pipeline by comparing performance across different controller families against ground-truth data. (a) shows joystick R^2 correlation scores (averaged for both left and right joysticks) with an overall average of 0.84. (b) shows button frame accuracy with an overall average of 0.96.

are generated with a diffusion transformer (DiT) (Peebles et al., 2023) that outputs multiple actions per forward pass. Noisy action chunks are first encoded by an MLP into one action token per timestep, then processed through several DiT blocks consisting of alternating self-attention and cross-attention layers. Cross-attention conditions action generation on the encoded frame tokens. The final action tokens are decoded into continuous action vectors using an MLP applied independently across the time dimension.

Design choices. Although the model can condition on multiple frames, we find no benefit from using more than one past frame, even with increased temporal gaps. This is likely because the initial state of these action games already provides sufficient context to elicit the appropriate behavior. We instead use a single context frame and generate 16-action chunks, which improves temporal consistency compared to single-action generation.

Training and inference. We train NitroGen using the standard conditional flow-matching objective (Lipman et al., 2022; Black et al., 2024), applied to 16-action chunks, with one 256×256 frame of context. Inference follows the corresponding denoising process with $k = 16$ steps.

During training, we apply the following image augmentations: random brightness, contrast, saturation and hue, random rotation between -5 and 5 degrees, and random crops. We train all models using AdamW (Kingma et al., 2017; Loshchilov et al., 2019) optimizer with a weight decay of 0.001. We use a warmup-stable-decay (WSD) schedule (Wen et al., 2024), which allows us to train for longer without a fixed

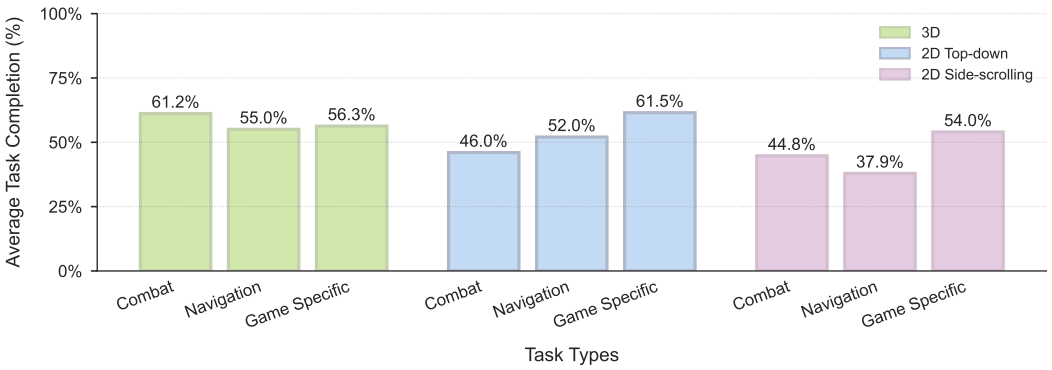


Figure 3.6: NitroGen 500M pre-training results across different games. We evaluate NitroGen after behavior-cloning pre-training. The model is not fine-tuned for specific games. For each game, we measure the average task completion rate on 3 tasks with 5 rollouts per task. Despite being trained on a very noisy internet dataset, NitroGen is able to perform non-trivial tasks over games with different visual styles (3D, 2D top-down, 2D side-scrolling) and genres (platformer, action-RPG, roguelike, etc.).

training budget, with a constant learning rate phase of 0.0001. Following Peebles et al., 2023, we maintain an exponential moving average (EMA) of model weights during training with a decay of 0.9999. All our results are obtained with the EMA weights, which we find consistently outperform the non-EMA weights.

3.4 Experiments

Performance of the gamepad action extraction model. To evaluate our action extraction pipeline, we construct a benchmark dataset by recording gameplay from six video games using OBS[4], with randomized opacity, gamepad size, and gamepad type to mimic real-world conditions. We record ground-truth controller inputs at each frame and compare them with the extracted actions. We measure joystick accuracy with the R^2 score and button accuracy per frame. As shown in Figure 3.5, we achieve an average R^2 of 0.84 for joystick positions and an average button accuracy of 0.96 across the most popular controller families.

NitroGen demonstrates strong capabilities across a wide range of games. We train a single model on the entire dataset from Section 3.3. Without further fine-tuning, NitroGen achieves non-trivial success rates across many games and tasks. Figure 3.6 summarizes the main results. We observe that NitroGen performs well both on tasks that can be memorized and on tasks that require zero-shot

[4]Open Broadcaster Software: `https://obsproject.com/`; Input recording tool: `https://github.com/loicmagne/input-rec`

generalization. For example, some games feature fixed layouts that the model may have partially encountered during training, while others employ procedural generation that ensures each playthrough is unique. We do not find significant differences in performance between these two categories, suggesting that NitroGen can both leverage memorization and adapt to unseen scenarios.

This result validates that it is possible to train a robust policy using only noisy internet-scale data. The dataset includes several sources of noise that could hinder training: **(a) actions are not strictly ground truth**, since input overlay software introduces small delays, and parsing adds further inaccuracies; **(b) video frames often contain creator-specific artifacts** such as livestream chats, subscribe prompts, or progress trackers; and **(c) controller configurations vary across players**, differences in sensitivity settings or custom button mappings can change the semantic meaning of the same input. Despite these challenges, Figure 3.6 shows that large-scale pre-training yields a robust multi-game policy.

NitroGen pre-training improves downstream fine-tuning on unseen environments. We evaluate transfer learning by pre-training NitroGen on the full dataset except for a held-out game, then fine-tuning on this game with a limited amount of data. We compare this fine-tuned model with an identical architecture trained from scratch using the same data and compute budget. Results are shown in Figure 3.7. We study two representative games with different visual styles and genres: an isometric roguelike and a 3D action-RPG.

The effectiveness of pre-training varies by game type and task category. Across different data quantities, fine-tuning achieves an average relative improvement of 10% on the isometric roguelike, whereas the 3D action-RPG shows a 25% average relative improvement. This difference likely stems from better representation of 3D action-RPGs in the training distribution, while the isometric roguelike has gameplay mechanics and visual style that are less common in the training data.

Furthermore, pre-training benefits are not uniform across task types. On the 3D action-RPG, generic tasks such as combat (52% relative improvement) and navigation (25% relative improvement) benefit substantially from pre-training, while game-specific tasks show only marginal gains (5% relative improvement). This suggests that NitroGen effectively learns transferable skills for common gameplay patterns, but game-specific mechanics still require targeted training on the new environment.

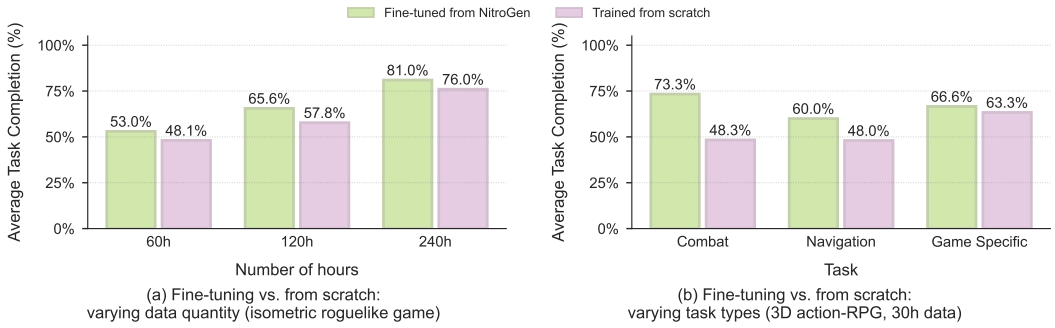


Figure 3.7: Post-training experiments: NitroGen pre-training improves downstream agents in unseen environments. We pre-train NitroGen on the dataset described in Section 3.3, holding out one game. We then fine-tune the pre-trained checkpoint on the held-out game and compare the results with a model trained from scratch using the same architecture, data and compute budget. (a) When varying data quantity, task-completion rate scales with dataset size, and fine-tuning achieves on average a 10% relative improvement in task-completion rate. (b) When varying task type in the low-data regime (30h), fine-tuning achieves up to 52% relative improvement in task-completion rate.

3.5 Limitations and Future Work

Design limitations. NitroGen is limited to being a fast-reacting system-1 sensory model. It cannot plan over long horizons or follow language instructions; the model only reacts to the short context it sees. We develop NitroGen aiming for it to serve as a foundation for future generalist agent development, where post-training for language-following and reinforcement learning can be applied to enhance planning capabilities and improve success rates.

Dataset bias. While diverse, our data collection method still restricts the types of games included in our dataset. The data distribution of the NitroGen dataset is biased toward action games (Figure 3.3), and games that are typically played with a gamepad. Keyboard-only games or those that involve complex manipulation are less represented in the dataset. This bias may limit the agent's ability to generalize to genres like strategy or simulation games that rely more on planning and keyboard input.

3.6 Conclusion

In this work, we introduce NitroGen, an approach to scale up foundation pre-training for video game agents and demonstrate how internet pre-training can yield a generalist policy. We leverage a new source of publicly available data to build an internet-scale video-action dataset, and empirically demonstrate its effectiveness by successfully

training a multi-game policy. NitroGen shows positive signs of generalization in fine-tuning experiments. By lowering the barrier to train agents on new environments, NitroGen serves as a starting point to develop more powerful and general-purpose agents.

Chapter 4

LARGE LANGUAGE MODELS FOR OPEN-ENDED AGENT PLANNING

This chapter is based on the paper:

Wang, Guanzhi, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar (2024). Voyager: An Open-Ended Embodied Agent With Large Language Models. In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: https://openreview.net/forum?id=ehfRiF0R3a.

4.1 Introduction

Building generally capable embodied agents that continuously explore, plan, and develop new skills in open-ended worlds is a grand challenge for the AI community (Kolve et al., 2017; Savva et al., 2019; Zhu et al., 2020c; Xia et al., 2019; Shen et al., 2020). Classical approaches employ reinforcement learning (RL) (Kober et al., 2013; Arulkumaran et al., 2017) and imitation learning (Baker et al., 2022; Team et al., 2021a; Vinyals et al., 2019a) that operate on primitive actions, which could be challenging for systematic exploration (Ecoffet et al., 2019; Huizinga et al., 2022; Wang et al., 2020; Kanitscheider et al., 2021; Dennis et al., 2020), interpretability (Liang et al., 2023; Sun et al., 2020; Zhao et al., 2021), and generalization (Jiang et al., 2023; Shridhar et al., 2021; Fan et al., 2021). Recent advances in large language model (LLM) based agents harness the world knowledge encapsulated in pre-trained LLMs to generate consistent action plans or executable policies (Liang et al., 2023; Singh et al., 2023; Jiang et al., 2023). They are applied to embodied tasks like games and robotics (Fan et al., 2022; Zeng et al., 2022; Ahn et al., 2022; Huang et al., 2022b; Huang et al., 2022a), as well as NLP tasks without embodiment (AutoGPT, 2023; Yao et al., 2022; Shinn et al., 2023). However, these agents are not lifelong learners that can progressively acquire, update, accumulate, and transfer knowledge over extended time spans (Parisi et al., 2019; Wang et al., 2023c).

Let us consider Minecraft as an example. Unlike most other games studied in AI (Mnih et al., 2013; OpenAI et al., 2019; Vinyals et al., 2019a), Minecraft does not impose a predefined end goal or a fixed storyline but rather provides a unique playground with endless possibilities (Fan et al., 2022). Minecraft requires players to

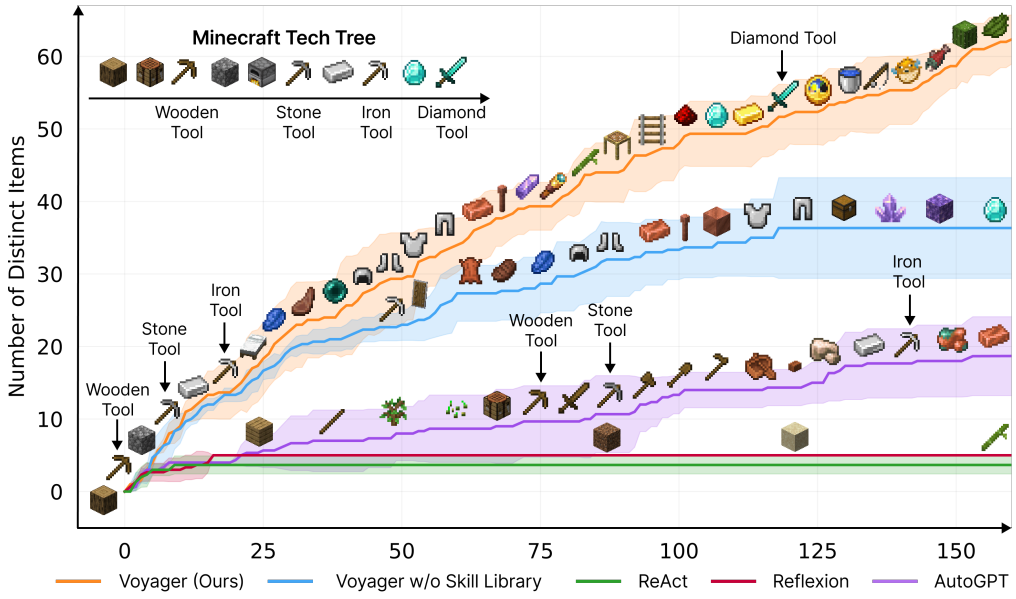


Figure 4.1: VOYAGER discovers new Minecraft items and skills continually by self-driven exploration, significantly outperforming the baselines. X-axis denotes the number of prompting iterations.

explore vast, procedurally generated 3D terrains and unlock a tech tree using gathered resources. Human players typically start by learning the basics, such as mining wood and cooking food, before advancing to more complex tasks like combating monsters and crafting diamond tools. We argue that an effective lifelong learning agent should have similar capabilities as human players: (1) **propose suitable tasks** based on its current skill level and world state, e.g., learn to harvest sand and cactus before iron if it finds itself in a desert rather than a forest; (2) **refine skills** based on environmental feedback and **commit mastered skills to memory** for future reuse in similar situations (e.g., fighting zombies is similar to fighting spiders); (3) **continually explore the world** and seek out new tasks in a self-driven manner.

Towards these goals, we introduce VOYAGER, the first *LLM-powered embodied lifelong learning agent* to drive exploration, master a wide range of skills, and make new discoveries continually without human intervention in Minecraft. VOYAGER is made possible through three key modules (Fig. 4.2): 1) an **automatic curriculum** that maximizes exploration; 2) a **skill library** for storing and retrieving complex behaviors; and 3) a new **iterative prompting mechanism** that generates executable code for embodied control. We opt to use code as the action space instead of low-level motor commands because programs can naturally represent temporally extended and compositional actions (Liang et al., 2023; Singh et al., 2023), which are essential

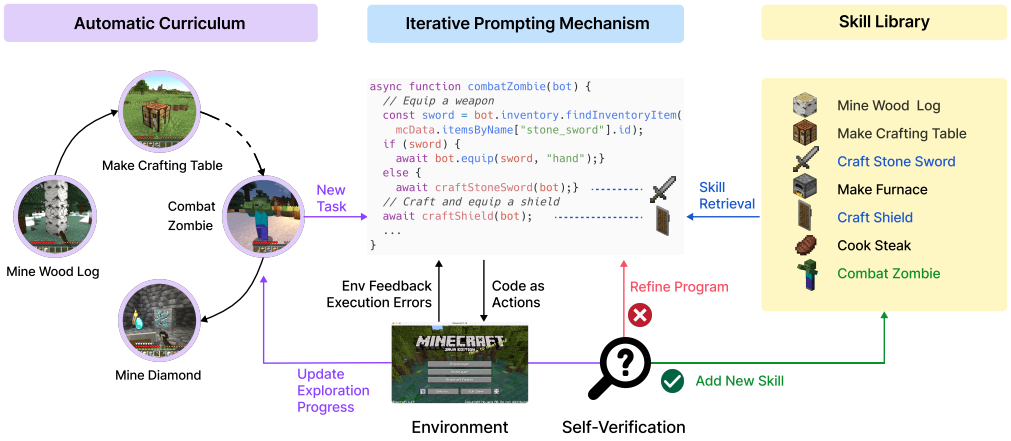


Figure 4.2: VOYAGER consists of three key components: an automatic curriculum for open-ended exploration, a skill library for increasingly complex behaviors, and an iterative prompting mechanism that uses code as action space.

for many long-horizon tasks in Minecraft. VOYAGER interacts with a blackbox LLM (GPT-4 (OpenAI, 2023)) through prompting and in-context learning (Wei et al., 2022a; Brown et al., 2020; Raffel et al., 2020). Our approach bypasses the need for model parameter access and explicit gradient-based training or finetuning.

More specifically, VOYAGER attempts to solve progressively harder tasks proposed by the **automatic curriculum**, which takes into account the exploration progress and the agent's state. The curriculum is generated by GPT-4 based on the overarching goal of "discovering as many diverse things as possible". This approach can be perceived as an in-context form of *novelty search* (Eysenbach et al., 2019; Conti et al., 2018). VOYAGER incrementally builds a **skill library** by storing the action programs that help solve a task successfully. Each program is indexed by the embedding of its description, which can be retrieved in similar situations in the future. Complex skills can be synthesized by *composing* simpler programs, which compounds VOYAGER's capabilities rapidly over time and alleviates catastrophic forgetting in other continual learning methods (Parisi et al., 2019; Wang et al., 2023c).

However, LLMs struggle to produce the correct action code consistently in one shot (Chen et al., 2021c). To address this challenge, we propose an **iterative prompting mechanism** that: (1) executes the generated program to obtain observations from the Minecraft simulation (such as inventory listing and nearby creatures) and error trace from the code interpreter (if any); (2) incorporates the feedback into GPT-4's prompt for another round of code refinement; and (3) repeats the process until a self-verification module confirms the task completion, at which

point we commit the program to the skill library (e.g., `craftStoneShovel()` and `combatZombieWithSword()`) and query the automatic curriculum for the next milestone (Fig. 4.2).

Empirically, VOYAGER demonstrates strong **in-context lifelong learning** capabilities. It can construct an ever-growing skill library of action programs that are reusable, interpretable, and generalizable to novel tasks. We evaluate VOYAGER systematically against other LLM-based agent techniques (e.g., ReAct (Yao et al., 2022), Reflexion (Shinn et al., 2023), AutoGPT (AutoGPT, 2023)) in MineDojo (Fan et al., 2022), an open-source Minecraft AI framework. VOYAGER outperforms prior SOTA by obtaining $3.3\times$ more unique items, unlocking key tech tree milestones up to $15.3\times$ faster, and traversing $2.3\times$ longer distances. We further demonstrate that VOYAGER is able to utilize the learned skill library in a new Minecraft world to solve novel tasks from scratch, while other methods struggle to generalize.

4.2 Related Work

Decision-making Agents in Minecraft. Minecraft is an open-ended 3D world with incredibly flexible game mechanics supporting a broad spectrum of activities. Built upon notable Minecraft benchmarks (Fan et al., 2022; Guss et al., 2019a; Guss et al., 2019b; Guss et al., 2021; Kanervisto et al., 2022; Johnson et al., 2016), Minecraft learning algorithms can be divided into two categories: 1) Low-level controller: Many prior efforts leverage hierarchical reinforcement learning to learn from human demonstrations (Lin et al., 2021b; Mao et al., 2021; Skrynnik et al., 2021). Kanitscheider et al. (Kanitscheider et al., 2021) design a curriculum based on success rates, but its objectives are limited to curated items. MineDojo (Fan et al., 2022) and VPT (Baker et al., 2022) utilize YouTube videos for large-scale pre-training. DreamerV3 (Hafner et al., 2023), on the other hand, learns a world model to explore the environment and collect diamonds. 2) High-level planner: Volum et al. (Volum et al., 2022) leverage few-shot prompting with Codex (Chen et al., 2021c) to generate executable policies, but they require additional human interaction. Recent works leverage LLMs as a high-level planner in Minecraft by decomposing a high-level task into several subgoals following Minecraft recipes (Wang et al., 2024b; Nottingham et al., 2023; Yuan et al., 2023), thus lacking full exploration flexibility. Like these latter works, VOYAGER also uses LLMs as a high-level planner by prompting GPT-4 and utilizes Mineflayer (PrismarineJS, 2013) as a low-level controller following Volum et al. (Volum et al., 2022). Unlike prior works, VOYAGER employs an automatic curriculum that unfolds in a bottom-up manner, driven by curiosity, and therefore enables open-ended exploration.

Large Language Models for Agent Planning. Inspired by the strong emergent capabilities of LLMs, such as zero-shot prompting and complex reasoning (Bommasani et al., 2021; Brown et al., 2020; Raffel et al., 2020; Wei et al., 2022a; Chowdhery et al., 2022; Chung et al., 2022), embodied agent research (Duan et al., 2022; Batra et al., 2020; Ravichandar et al., 2020; Collins et al., 2021) has witnessed a significant increase in the utilization of LLMs for planning purposes. Recent efforts can be roughly classified into two groups. 1) Large language models for robot learning: Many prior works apply LLMs to generate subgoals for robot planning (Huang et al., 2022a; Huang et al., 2022a; Ahn et al., 2022; Min et al., 2021; Blukis et al., 2021). Inner Monologue (Huang et al., 2022b) incorporates environment feedback for robot planning with LLMs. Code as Policies (Liang et al., 2023) and ProgPrompt (Singh et al., 2023) directly leverage LLMs to generate executable robot policies. VIMA (Jiang et al., 2023) and PaLM-E (Driess et al., 2023) fine-tune pre-trained LLMs to support multimodal prompts. 2) Large language models for text agents: ReAct (Yao et al., 2022) leverages chain-of-thought prompting (Wei et al., 2022b) and generates both reasoning traces and task-specific actions with LLMs. Reflexion (Shinn et al., 2023) is built upon ReAct (Yao et al., 2022) with self-reflection to enhance reasoning. AutoGPT (AutoGPT, 2023) is a popular tool that automates NLP tasks by crafting a curriculum of multiple subgoals for completing a high-level goal while incorporating ReAct (Yao et al., 2022)'s reasoning and acting loops. DERA (Nair et al., 2023) frames a task as a dialogue between two GPT-4 (OpenAI, 2023) agents. Generative Agents (Park et al., 2023) leverages ChatGPT (*Introducing ChatGPT* 2022) to simulate human behaviors by storing agents' experiences as memories and retrieving those for planning, but its agent actions are not executable. SPRING (Wu et al., 2023c) is a concurrent work that uses GPT-4 to extract game mechanics from game manuals, based on which it answers questions arranged in a directed acyclic graph and predicts the next action. All these works lack a skill library for developing more complex behaviors, which are crucial components for the success of VOYAGER in lifelong learning.

Code Generation with Execution. Code generation has been a longstanding challenge in NLP (Chen et al., 2021c; Nijkamp et al., 2022; Le et al., 2022; Chowdhery et al., 2022; Brown et al., 2020), with various works leveraging execution results to improve program synthesis. Execution-guided approaches leverage intermediate execution outcomes to guide program search (Chen et al., 2019; Chen et al., 2021d; Ellis et al., 2019). Another line of research utilizes majority voting to choose candidates based on their execution performance (Li et al., 2022b; Cobbe et al., 2021). Additionally,

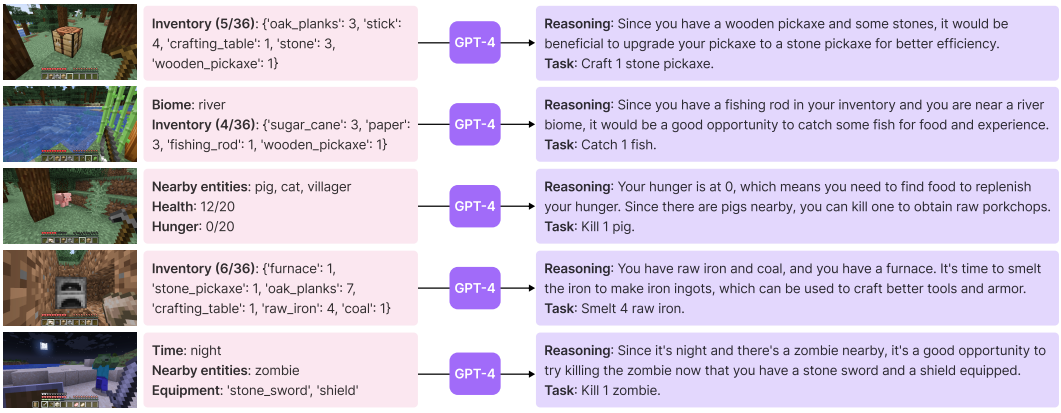


Figure 4.3: Tasks proposed by the automatic curriculum. We only display the partial prompt for brevity.

LEVER (Ni et al., 2023) trains a verifier to distinguish and reject incorrect programs based on execution results. CLAIRIFY (Skreta et al., 2023), on the other hand, generates code for planning chemistry experiments and makes use of a rule-based verifier to iteratively provide error feedback to LLMs. VOYAGER distinguishes itself from these works by integrating environment feedback, execution errors, and self-verification (to assess task success) into an iterative prompting mechanism for embodied control.

4.3 Method

VOYAGER consists of three novel components: (1) an automatic curriculum (Sec. 4.3) that suggests objectives for open-ended exploration, (2) a skill library (Sec. 4.3) for developing increasingly complex behaviors, and (3) an iterative prompting mechanism (Sec. 4.3) that generates executable code for embodied control.

Automatic Curriculum

Embodied agents encounter a variety of objectives with different complexity levels in open-ended environments. An automatic curriculum offers numerous benefits for open-ended exploration, ensuring a challenging but manageable learning process, fostering curiosity-driven intrinsic motivation for agents to learn and explore, and encouraging the development of general and flexible problem-solving strategies (Wang et al., 2019; Portelas et al., 2020; Forestier et al., 2022). Our automatic curriculum capitalizes on the internet-scale knowledge contained within GPT-4 by prompting it to provide a steady stream of new tasks or challenges. The curriculum evolves bottom-up, allowing for considerable adaptability and responsiveness to the exploration progress and the agent's current state (Fig. 4.3). As VOYAGER progresses to harder self-driven goals, it naturally learns a variety of skills, such as "mining a diamond."

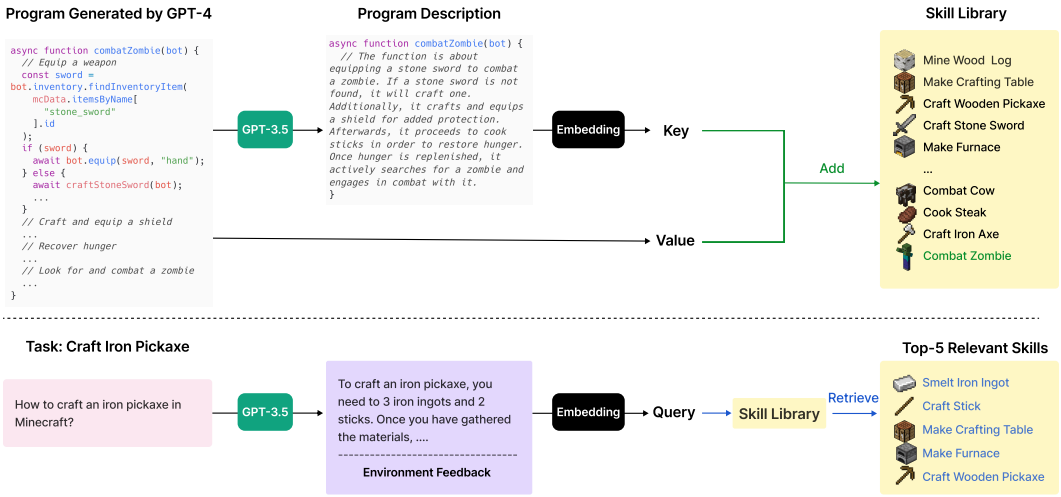


Figure 4.4: Skill library. Top: Adding a new skill. Each time GPT-4 generates and verifies a new skill, we add it to the skill library, represented by a vector database. The key is the embedding vector of the program description (generated by GPT-3.5), while the value is the program itself. Bottom: Skill retrieval. When faced with a new task proposed by the automatic curriculum, we first leverage GPT-3.5 to generate a general suggestion for solving the task, which is combined with environment feedback as the query context. Subsequently, we perform querying to identify the top-5 relevant skills.

The input prompt to GPT-4 consists of several components:

(1) **Directives encouraging diverse behaviors and imposing constraints**, such as "`My ultimate goal is to discover as many diverse things as possible ... The next task should not be too hard since I may not have the necessary resources or have learned enough skills to complete it yet.`";

(2) **The agent's current state**, including inventory, equipment, nearby blocks and entities, biome, time, health and hunger bars, and position;

(3) **Previously completed and failed tasks**, reflecting the agent's current exploration progress and capabilities frontier;

(4) **Additional context**: We also leverage GPT-3.5 to self-ask questions based on the agent's current state and exploration progress and self-answer questions. We opt to use GPT-3.5 instead of GPT-4 for standard NLP tasks due to budgetary considerations.

Skill Library

With the automatic curriculum consistently proposing increasingly complex tasks, it is essential to have a skill library that serves as a basis for learning and evolution. Inspired by the generality, interpretability, and universality of programs (Ellis et al., 2020), we represent each skill with executable code that scaffolds temporally extended actions for completing a specific task proposed by the automatic curriculum.

The input prompt to GPT-4 consists of the following components:

(1) **Guidelines for code generation**, such as "`Your function will be reused for building more complex functions. Therefore, you should make it generic and reusable.`";

(2) **Control primitive APIs, and relevant skills** retrieved from the skill library, which are crucial for in-context learning (Wei et al., 2022a; Brown et al., 2020; Raffel et al., 2020) to work well;

(3) **The generated code from the last round, environment feedback, execution errors, and critique**, based on which GPT-4 can self-improve (Sec. 4.3);

(4) **The agent's current state**, including inventory, equipment, nearby blocks and entities, biome, time, health and hunger bars, and position;

(5) **Chain-of-thought prompting** (Wei et al., 2022b) to do reasoning before code generation.

We iteratively refine the program through a novel iterative prompting mechanism (Sec. 4.3), incorporate it into the skill library as a new skill, and index it by the embedding of its description (Fig. 4.4, top). For skill retrieval, we query the skill library with the embedding of self-generated task plans and environment feedback (Fig. 4.4, bottom). By continuously expanding and refining the skill library, VOYAGER can learn, adapt, and excel in a wide spectrum of tasks, consistently pushing the boundaries of its capabilities in the open world.

Iterative Prompting Mechanism

We introduce an iterative prompting mechanism for self-improvement through three types of feedback:

(1) **Environment feedback**, which illustrates the intermediate progress of program execution (Fig. 4.5, left). For example, "`I cannot make an iron`

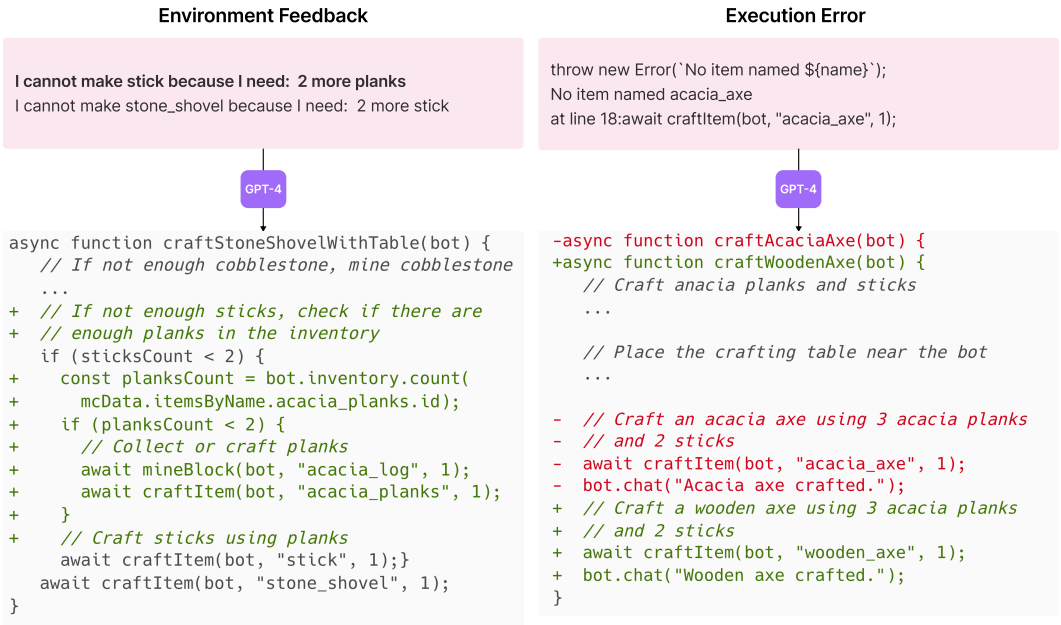


Figure 4.5: Iterative prompting. Left: Environment feedback. GPT-4 realizes it needs 2 more planks before crafting sticks. Right: Execution error. GPT-4 realizes it should craft a wooden axe instead of an acacia axe since there is no acacia axe in Minecraft. We only display the partial prompt for brevity.

`chestplate because I need: 7 more iron ingots`" highlights the cause of failure in crafting an iron chestplate. We use `bot.chat()` inside control primitive APIs to generate environment feedback and prompt GPT-4 to use this function as well during code generation;

(2) **Execution errors** from the program interpreter that reveal any invalid operations or syntax errors in programs, which are valuable for bug fixing (Fig. 4.5, right);

(3) **Self-verification for checking task success.** Instead of manually coding success checkers for each new task proposed by the automatic curriculum, we instantiate another GPT-4 agent for self-verification. By providing VOYAGER's current state and the task to GPT-4, we ask it to act as a critic (Mnih et al., 2016; Schulman et al., 2017; Lillicrap et al., 2016) and inform us whether the program achieves the task. In addition, if the task fails, it provides a critique by suggesting how to complete the task (Fig. 4.6). Hence, our self-verification is more comprehensive than self-reflection (Shinn et al., 2023) by both checking success and reflecting on mistakes.

Figure 4.6: Self-verification examples. We only display the partial prompt for brevity.

During each round of code generation, we execute the generated program to obtain environment feedback and execution errors from the code interpreter, which are incorporated into GPT-4's prompt for the next round of code refinement. This iterative process repeats until self-verification validates the task's completion, at which point we add this new skill to the skill library and ask the automatic curriculum for a new objective (Fig. 4.2). If the agent gets stuck after 4 rounds of code generation, then we query the curriculum for another task. This iterative prompting approach significantly improves program synthesis for embodied control, enabling VOYAGER to continuously acquire diverse skills without human intervention.

4.4 Experiments

Experimental Setup

We leverage OpenAI's `gpt-4-0314` (OpenAI, 2023) and `gpt-3.5-turbo-0301` (*Introducing ChatGPT* 2022) APIs for text completion, along with `text-embedding-ada-002` (*New and Improved Embedding Model* 2022) API for text embedding. We set all temperatures to 0 except for the automatic curriculum, which uses temperature $= 0.1$ to encourage task diversity. Our simulation environment is built on top of MineDojo (Fan et al., 2022) and leverages Mineflayer (PrismarineJS, 2013) JavaScript APIs for motor controls.

Baselines

Because there is no LLM-based agents that work out of the box for Minecraft, we make our best effort to select a number of representative algorithms as baselines. These methods are originally designed only for NLP tasks without embodiment, therefore we have to re-interpret them to be executable in MineDojo and compatible

with our experimental setting:

ReAct (Yao et al., 2022) uses chain-of-thought prompting (Wei et al., 2022b) by generating both reasoning traces and action plans with LLMs. We provide it with our environment feedback and the agent states as observations.

Reflexion (Shinn et al., 2023) is built on top of ReAct (Yao et al., 2022) with self-reflection to infer more intuitive future actions. We provide it with execution errors and our self-verification module.

AutoGPT (AutoGPT, 2023) is a popular software tool that automates NLP tasks by decomposing a high-level goal into multiple subgoals and executing them in a ReAct-style loop. We re-implement AutoGPT by using GPT-4 to do task decomposition and provide it with the agent states, environment feedback, and execution errors as observations for subgoal execution. Compared with VOYAGER, AutoGPT lacks the skill library for accumulating knowledge, self-verification for assessing task success, and automatic curriculum for open-ended exploration.

Note that we do not directly compare with prior methods that take Minecraft screen pixels as input and output low-level controls (Nottingham et al., 2023; Cai et al., 2023; Wang et al., 2024b). It would not be an apple-to-apple comparison, because we rely on the high-level Mineflayer (PrismarineJS, 2013) API to control the agent. Our work's focus is on pushing the limits of GPT-4 for lifelong embodied agent learning, rather than solving the 3D perception or sensorimotor control problems. VOYAGER is orthogonal and can be combined with gradient-based approaches like VPT (Baker et al., 2022) as long as the controller provides a code API.

Evaluation Results

We systematically evaluate VOYAGER and baselines on their exploration performance, tech tree mastery, map coverage, and zero-shot generalization capability to novel tasks in a new world.

Significantly better exploration. Results of exploration performance are shown in Fig. 4.1. VOYAGER's superiority is evident in its ability to consistently make new strides, discovering 63 unique items within 160 prompting iterations, $3.3\times$ many novel items compared to its counterparts. On the other hand, AutoGPT lags considerably in discovering new items, while ReAct and Reflexion struggle to make significant progress, given the abstract nature of the open-ended exploration goal that is challenging to execute without an appropriate curriculum.

Table 4.1: Tech tree mastery. Fractions indicate the number of successful trials out of three total runs. 0/3 means the method fails to unlock a level of the tech tree within the maximal prompting iterations (160). Numbers are prompting iterations averaged over three trials. The fewer the iterations, the more efficient the method.

Method	Wooden Tool	Stone Tool	Iron Tool	Diamond Tool
ReAct Yao et al., 2022	N/A (0/3)	N/A (0/3)	N/A (0/3)	N/A (0/3)
Reflexion Shinn et al., 2023	N/A (0/3)	N/A (0/3)	N/A (0/3)	N/A (0/3)
AutoGPT AutoGPT, 2023	92 ± 72 **(3/3)**	94 ± 72 **(3/3)**	135 ± 103 **(3/3)**	N/A (0/3)
VOYAGER w/o Skill Library	$\mathbf{7 \pm 2}$ **(3/3)**	$\mathbf{9 \pm 4}$ **(3/3)**	29 ± 11 **(3/3)**	N/A (0/3)
VOYAGER (Ours)	$\mathbf{6 \pm 2}$ **(3/3)**	$\mathbf{11 \pm 2}$ **(3/3)**	$\mathbf{21 \pm 7}$ **(3/3)**	**102 (1/3)**

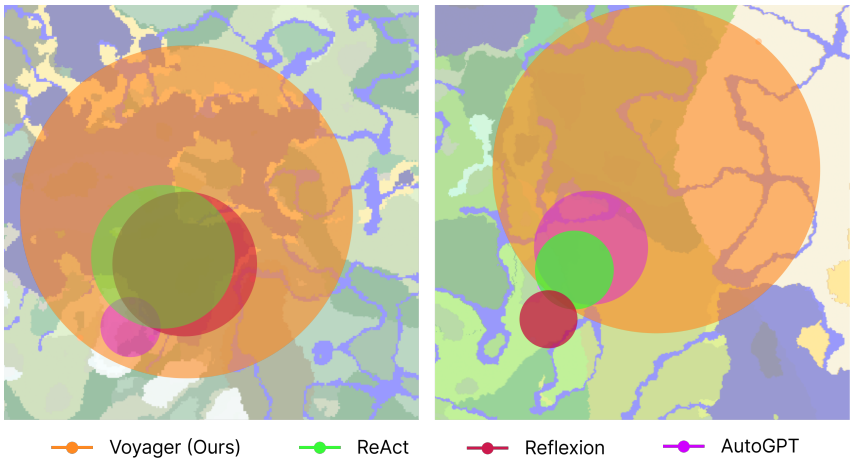


Figure 4.7: Map coverage: bird's eye views of Minecraft maps. VOYAGER is able to traverse $2.3\times$ longer distances compared to baselines while crossing diverse terrains.

Consistent tech tree mastery. The Minecraft tech tree tests the agent's ability to craft and use a hierarchy of tools. Progressing through this tree (wooden tool $\rightarrow$ stone tool $\rightarrow$ iron tool $\rightarrow$ diamond tool) requires the agent to master systematic and compositional skills. Compared with baselines, VOYAGER unlocks the wooden level $15.3\times$ faster (in terms of the prompting iterations), the stone level $8.5\times$ faster, the iron level $6.4\times$ faster, and VOYAGER is the only one to unlock the diamond level of the tech tree (Fig. 4.2 and Table. 4.1). This underscores the effectiveness of the automatic curriculum, which consistently presents challenges of suitable complexity to facilitate the agent's progress.

Extensive map traversal. VOYAGER is able to navigate distances $2.3\times$ longer compared to baselines by traversing a variety of terrains, while the baseline agents often find themselves confined to local areas, which significantly hampers their capacity to discover new knowledge (Fig. 4.7).

Efficient zero-shot generalization to unseen tasks. To evaluate zero-shot gener-

Table 4.2: Zero-shot generalization to unseen tasks. Fractions indicate the number of successful trials out of three total attempts. 0/3 means the method fails to solve the task within the maximal prompting iterations (50). Numbers are prompting iterations averaged over three trials. The fewer the iterations, the more efficient the method.

Method	Diamond Pickaxe	Golden Sword	Lava Bucket	Compass
ReAct Yao et al., 2022	N/A (0/3)	N/A (0/3)	N/A (0/3)	N/A (0/3)
Reflexion Shinn et al., 2023	N/A (0/3)	N/A (0/3)	N/A (0/3)	N/A (0/3)
AutoGPT AutoGPT, 2023	N/A (0/3)	N/A (0/3)	N/A (0/3)	N/A (0/3)
AutoGPT AutoGPT, 2023 w/ Our Skill Library	39 (1/3)	30 (1/3)	N/A (0/3)	30 (2/3)
VOYAGER w/o Skill Library	36 (2/3)	30 ± 9 **(3/3)**	27 ± 9 **(3/3)**	26 ± 3 **(3/3)**
VOYAGER (Ours)	$\mathbf{19 \pm 3}$ **(3/3)**	$\mathbf{18 \pm 7}$ **(3/3)**	$\mathbf{21 \pm 5}$ **(3/3)**	$\mathbf{18 \pm 2}$ **(3/3)**

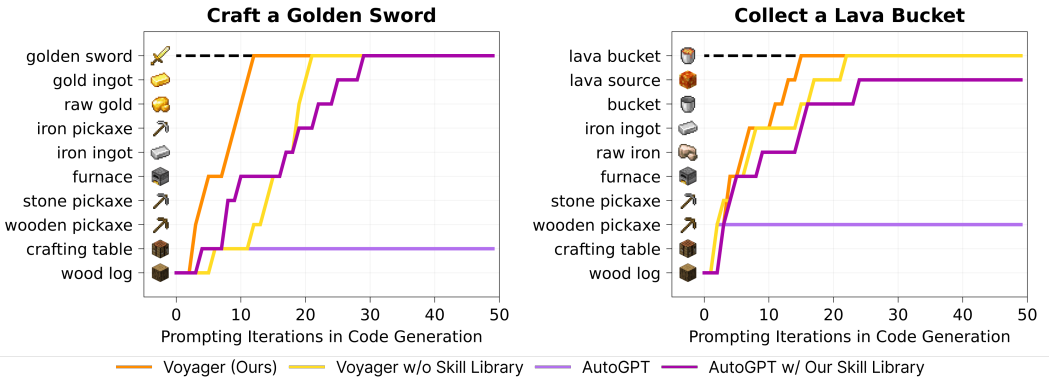


Figure 4.8: Zero-shot generalization to unseen tasks. We visualize the intermediate progress of each method on two tasks. We do not plot ReAct and Reflexion since they do not make any meaningful progress.

alization, we clear the agent's inventory, reset it to a newly instantiated world, and test it with unseen tasks. For both VOYAGER and AutoGPT, we utilize GPT-4 to break down the task into a series of subgoals. Table. 4.2 and Fig. 4.8 show VOYAGER can consistently solve all the tasks, while baselines cannot solve any task within 50 prompting iterations. What's interesting to note is that our skill library constructed from lifelong learning not only enhances VOYAGER's performance but also gives a boost to AutoGPT. This demonstrates that the skill library serves as a versatile tool that can be readily employed by other methods, effectively acting as a plug-and-play asset to enhance performance.

Ablation Studies

We ablate 6 design choices (automatic curriculum, skill library, environment feedback, execution errors, self-verification, and GPT-4 for code generation) in VOYAGER and study their impact on exploration performance. Results are shown in Fig. 4.9. We highlight the key findings below:

- **Automatic curriculum is crucial for the agent's consistent progress.** The discovered item count drops by 93% if the curriculum is replaced with a random one, because certain tasks may be too challenging if attempted out of order. On the other hand, a manually designed curriculum requires significant Minecraft-specific expertise, and does not take into account the agent's live situation. It falls short in the experimental results compared to our automatic curriculum.

- **VOYAGER w/o skill library exhibits a tendency to plateau in the later stages.** This underscores the pivotal role that the skill library plays in VOYAGER. It helps create more complex actions and steadily pushes the agent's boundaries by encouraging new skills to be built upon older ones.

- **Self-verification is the most important among all the feedback types**. Removing the module leads to a significant drop (-73%) in the discovered item count. Self-verification serves as a critical mechanism to decide when to move on to a new task or reattempt a previously unsuccessful task.

- **GPT-4 significantly outperforms GPT-3.5 in code generation** and obtains $5.7\times$ more unique items, as GPT-4 exhibits a quantum leap in coding abilities. This finding corroborates recent studies in the literature (Bubeck et al., 2023; Liu et al., 2023c).

Multimodal Feedback from Humans

VOYAGER does not currently support visual perception, because the available version of GPT-4 API is text-only at the time of this writing. However, VOYAGER has the potential to be augmented by multimodal perception models (Liu et al., 2023b; Driess et al., 2023) to achieve more impressive tasks. We demonstrate that given human feedback, VOYAGER is able to construct complex 3D structures in Minecraft, such as a Nether Portal and a house (Fig. 4.10). There are two ways to integrate human feedback:

(1) Human as a critic (equivalent to VOYAGER's self-verification module): humans provide visual critique to VOYAGER, allowing it to modify the code from the previous round. This feedback is essential for correcting certain errors in the spatial details of a 3D structure that VOYAGER cannot perceive directly.

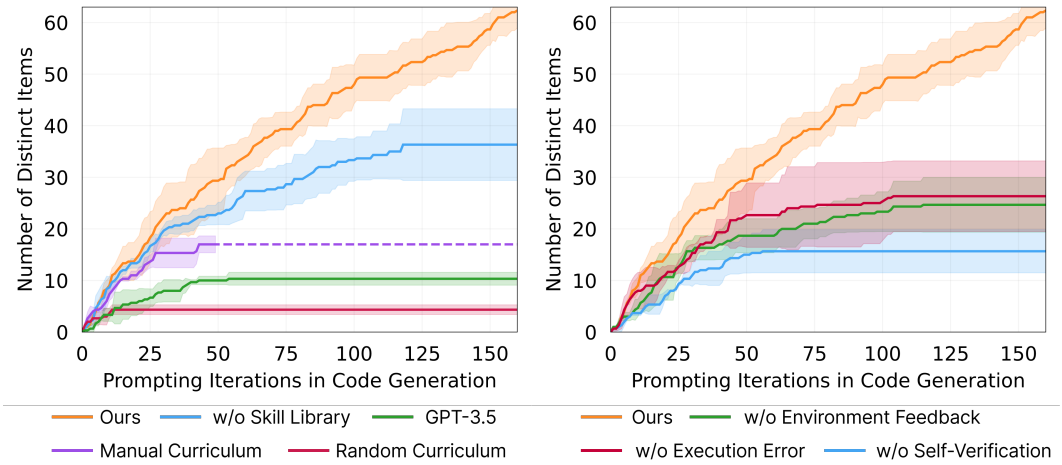


Figure 4.9: Ablations. Left: Ablation studies for the automatic curriculum, skill library, and GPT-4. GPT-3.5 means replacing GPT-4 with GPT-3.5 for code generation. VOYAGER outperforms all the alternatives, demonstrating the critical role of each component. Right: Ablation studies for the iterative prompting mechanism. VOYAGER surpasses all the other options, thereby highlighting the essential significance of each type of feedback in the iterative prompting mechanism.

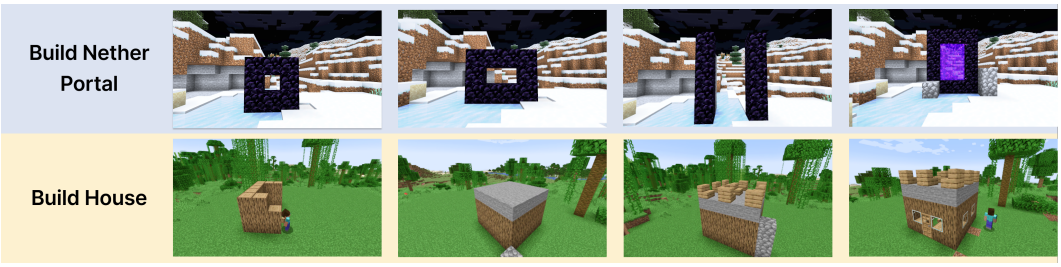


Figure 4.10: VOYAGER builds 3D structures with human feedback. The progress of building designs that integrate human input is demonstrated from left to right.

(2) Human as a curriculum (equivalent to VOYAGER's automatic curriculum module): humans break down a complex building task into smaller steps, guiding VOYAGER to complete them incrementally. This approach improves VOYAGER's ability to handle more sophisticated 3D construction tasks.

4.5 Limitations and Future Work

Cost. The GPT-4 API incurs significant costs. It is $15\times$ more expensive than GPT-3.5. Nevertheless, VOYAGER requires the quantum leap in code generation quality from GPT-4 (Fig. 4.9), which GPT-3.5 and open-source LLMs cannot provide (Touvron et al., 2023).

Inaccuracies. Despite the iterative prompting mechanism, there are still cases where the agent gets stuck and fails to generate the correct skill. The automatic curriculum

has the flexibility to reattempt this task at a later time. Occasionally, self-verification module may also fail, such as not recognizing spider string as a success signal of beating a spider.

Hallucinations. The automatic curriculum occasionally proposes unachievable tasks. For example, it may ask the agent to craft a “copper sword" or “copper chestplate", which are items that do not exist within the game. Hallucinations also occur during the code generation process. For instance, GPT-4 tends to use cobblestone as a fuel input, despite being an invalid fuel source in the game. Additionally, it may call functions absent in the provided control primitive APIs, leading to code execution errors.

We are confident that improvements in the GPT API models as well as novel techniques for finetuning open-source LLMs will overcome these limitations in the future.

4.6 Conclusion

In this work, we introduce VOYAGER, the first LLM-powered embodied lifelong learning agent, which leverages GPT-4 to explore the world continuously, develop increasingly sophisticated skills, and make new discoveries consistently without human intervention. VOYAGER exhibits superior performance in discovering novel items, unlocking the Minecraft tech tree, traversing diverse terrains, and applying its learned skill library to unseen tasks in a newly instantiated world. VOYAGER serves as a starting point to develop powerful generalist agents without tuning the model parameters.

4.7 Broader Impacts

Our research is conducted within Minecraft, a safe and harmless 3D video game environment. While VOYAGER is designed to be generally applicable to other domains, such as robotics, its application to physical robots would require additional attention and the implementation of safety constraints by humans to ensure responsible and secure deployment.

Chapter 5

LARGE LANGUAGE MODELS FOR ROBOT REWARD DESIGN

This chapter is based on the paper:

Ma, Yecheng Jason, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, and Anima Anandkumar (2024). Eureka: Human-Level Reward Design via Coding Large Language Models. In: *International Conference on Representation Learning* 2024. Ed. by B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, pp. 26516–26560. URL: https://proceedings.iclr.cc/paper_files/paper/2024/file/70c26937fbf3d4600b69a129031b66ec-Paper-Conference.pdf.

5.1 Introduction

Large Language Models (LLMs) have excelled as high-level semantic planners for robotics tasks (Ahn et al., 2022; Singh et al., 2023), but whether they can be used to learn complex low-level manipulation tasks, such as dexterous pen spinning, remains an open problem. Existing attempts require substantial domain expertise to construct task prompts or learn only simple skills, leaving a substantial gap in achieving human-level dexterity (Yu et al., 2023b; Brohan et al., 2023b).

On the other hand, reinforcement learning (RL) has achieved impressive results in dexterity (Andrychowicz et al., 2020; Handa et al., 2023) as well as many other domains-if the human designers can carefully construct reward functions that accurately codify and provide learning signals for the desired behavior; likewise, many real-world RL tasks admit sparse rewards that are difficult for learning, necessitating reward shaping that provides incremental learning signals. Despite their fundamental importance, reward functions are known to be notoriously difficult to design in practice (Russell et al., 1995; Sutton et al., 2018); a recent survey conducted finds 92% of polled reinforcement learning researchers and practitioners report manual trial-and-error reward design and 89% indicate that their designed rewards are sub-optimal (Booth et al., 2023) and lead to unintended behavior (Hadfield-Menell et al., 2017).

Given the paramount importance of reward design, we ask whether it is possible to develop a *universal* reward programming algorithm using state-of-the-art coding LLMs, such as GPT-4. Their remarkable abilities in code writing, zero-shot

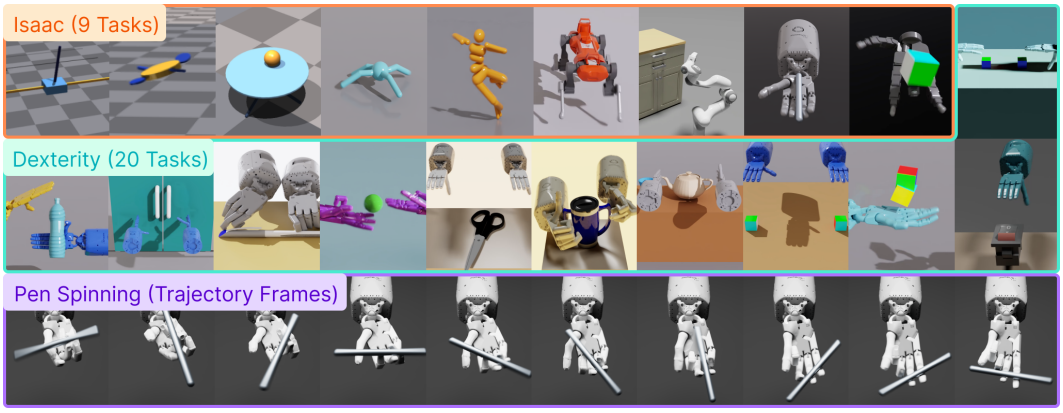


Figure 5.1: EUREKA generates human-level reward functions across diverse robots and tasks. Combined with curriculum learning, EUREKA for the first time, unlocks rapid pen-spinning capabilities on an anthropomorphic five-finger hand.

generation, and in-context learning have previously enabled effective programmatic agents (Shinn et al., 2023; Wang et al., 2023a). Ideally, this reward design algorithm should achieve human-level reward generation capabilities that scale to a broad spectrum of tasks, including dexterity, automate the tedious trial-and-error procedure without human supervision, and yet be compatible with human oversight to assure safety and alignment.

We introduce **E**volution-driven **U**niversal **RE**ward **K**it for **A**gent (EUREKA), a novel reward design algorithm powered by coding LLMs with the following contributions:

1. **Achieves human-level performance on reward design** across a diverse suite of 29 open-sourced RL environments that include 10 distinct robot morphologies, including quadruped, quadcopter, biped, manipulator, as well as several dexterous hands; see Fig. 5.1. Without any task-specific prompting or reward templates, EUREKA autonomously generates rewards that outperform expert human rewards on **83%** of the tasks and realizes an average normalized improvement of **52%**.

2. **Solves dexterous manipulation tasks that were previously not feasible by manual reward engineering**. We consider pen spinning, in which a five-finger hand needs to rapidly rotate a pen in pre-defined spinning configurations for as many cycles as possible. Combining EUREKA with curriculum learning, we demonstrate for the first time rapid pen spinning maneuvers on a simulated anthropomorphic Shadow Hand (see Figure 5.1 bottom).

3. **Enables a new *gradient-free* in-context learning approach to reinforcement learning from human feedback (RLHF)** that can generate more performant

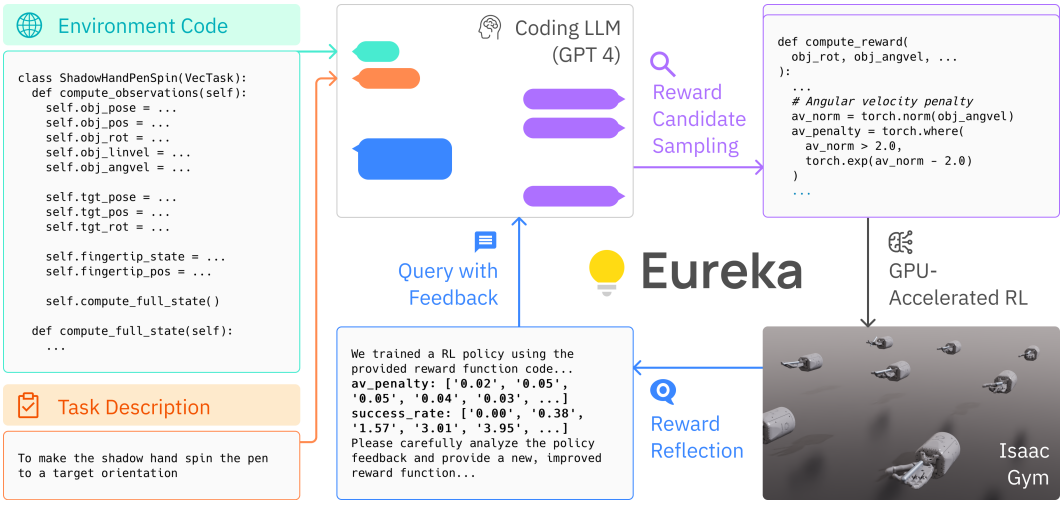


Figure 5.2: EUREKA takes unmodified environment source code and language task description as context to zero-shot generate executable reward functions from a coding LLM. Then, it iterates between reward sampling, GPU-accelerated reward evaluation, and reward reflection to progressively improve its reward outputs.

and human-aligned reward functions based on various forms of human inputs without model updating. We demonstrate that EUREKA can readily benefit from and improve upon existing human reward functions. Likewise, we showcase EUREKA's capability in using purely textual feedback to generate progressively more human-aligned reward functions.

Unlike prior work L2R on using LLMs to aid reward design (Yu et al., 2023b), EUREKA is completely free of task-specific prompts, reward templates, as well as few-shot examples. In our experiments, EUREKA significantly outperforms L2R due to its ability to generate free-form, expressive reward programs. EUREKA's generality is made possible through three key algorithmic design choices: environment as context, evolutionary search, and reward reflection. First, by taking the **environment source code as context**, EUREKA can zero-shot generate executable reward functions from the backbone coding LLM (GPT-4). Then, EUREKA substantially improves the quality of its rewards by performing **evolutionary search**, iteratively proposing batches of reward candidates and refining the most promising ones within the LLM context window. This in-context improvement is made effective via **reward reflection**, a textual summary of the reward quality based on policy training statistics that enables automated and targeted reward editing; see Fig. 5.3 for an example of EUREKA zero-shot reward as well as various improvements accumulated during its optimization. To ensure that EUREKA can scale up its reward search to maximum

potential, EUREKA evaluates intermediate rewards using GPU-accelerated distributed reinforcement learning on IsaacGym (Makoviychuk et al., 2021), which offers up to three orders of magnitude in policy learning speed, making EUREKA an extensive algorithm that scales naturally with more compute. See Fig. 5.2 for an overview. We are committed to open-sourcing all prompts, environments, and generated reward functions to promote further research on LLM-based reward design.

5.2 Related Work

Reward Design. Reward engineering is a long-standing challenge in reinforcement learning (Singh et al., 2009; Sutton et al., 2018). The most common reward design method is manual trial-and-error (Knox et al., 2023; Booth et al., 2023). Inverse reinforcement learning (IRL) infers reward functions from demonstrations (Abbeel et al., 2004; Ziebart et al., 2008; Ho et al., 2016), but it requires expensive expert data collection, which may not be available, and outputs non-interpretable black-box reward functions. Several prior works have studied automated reward search through evolutionary algorithms (Niekum et al., 2010; Chiang et al., 2019; Faust et al., 2019). These early attempts are limited to task-specific implementations of evolutionary algorithms that search over only parameters within provided reward templates. Recent works have also proposed using pretrained foundation models that can produce reward functions for new tasks (Ma et al., 2022; Ma et al., 2023; Fan et al., 2022; Du et al., 2023a; Karamcheti et al., 2023; Du et al., 2023b; Kwon et al., 2023). Most of these approaches output *scalar* rewards that lack interpretability and do not naturally admit the capability to improve or adapt rewards on-the-fly. In contrast, EUREKA adeptly generates free-form, white-box reward code and effectively in-context improves.

Code Large Language Models for Decision Making. Recent works have considered using coding LLMs (Austin et al., 2021; Chen et al., 2021c; Rozière et al., 2023) to generate grounded and structured programmatic output for decision making and robotics problems (Liang et al., 2023; Singh et al., 2023; Wang et al., 2023b; Huang et al., 2023a; Wang et al., 2023a; Liu et al., 2023a; Silver et al., 2023; Ding et al., 2023; Lin et al., 2023; Xie et al., 2023). However, most of these works rely on known motion primitives to carry out robot actions and do not apply to robot tasks that require low-level skill learning, such as dexterous manipulation. The closest to our work is a recent work (Yu et al., 2023b) that also explores using LLMs to aid reward design. Their approach, however, requires domain-specific task descriptions and reward templates.

5.3 Problem Setting and Definitions

The goal of reward design is to return a shaped reward function for a ground-truth reward function that may be difficult to optimize directly (e.g., sparse rewards); this ground-truth reward function may only be accessed via queries by the designer. We first introduce the formal definition from Singh et al. (2009), which we then adapt to the program synthesis setting, which we call *reward generation*.

Definition 5.3.1. (Reward Design Problem (Singh et al., 2009)) A *reward design problem* (RDP) is a tuple $P = \langle M, \mathcal{R}, \pi_M, F \rangle$, where $M = (S, A, T)$ is the *world model* with state space S, action space A, and transition function T. $\mathcal{R}$ is the space of reward functions; $\mathcal{A}_M(\cdot) : \mathcal{R} \to \Pi$ is a learning algorithm that outputs a policy $\pi : S \to \Delta(A)$ that optimizes reward $R \in \mathcal{R}$ in the resulting *Markov Decision Process* (MDP), (M, R); $F : \Pi \to \mathbb{R}$ is the *fitness* function that produces a scalar evaluation of any policy, which may only be accessed via policy queries (i.e., evaluate the policy using the ground truth reward function). In an RDP, the goal is to output a reward function $R \in \mathcal{R}$ such that the policy $\pi := \mathcal{A}_M(R)$ that optimizes R achieves the highest fitness score $F(\pi)$.

Reward Generation Problem. In our problem setting, every component within a RDP is specified via code. Then, given a string l that specifies the task, the objective of the reward generation problem is to output a reward function code R such that $F(\mathcal{A}_M(R))$ is maximized.

5.4 Method

EUREKA consists of three algorithmic components: 1) environment as context that enables zero-shot generation of executable rewards, 2) evolutionary search that iteratively proposes and refines reward candidates, and 3) reward reflection that enables fine-grained reward improvement. See Alg. 1 for pseudocode.

Environment as Context

Reward design requires the environment specification to be provided to the LLM. We propose directly feeding the raw environment source code (without the reward code, if exists) as context. Given that any reward function is a function over the environment's state and action variables, the only requirement in the source code is that it exposes these environment variables, which is easy to satisfy. In cases where the source code is not available, relevant state information can also be supplied via an API, for example. In practice, to ensure that the environment code fits within

Algorithm 1: EUREKA

Require: Task description l, environment code M,
coding LLM `LLM`, fitness function F, initial prompt `prompt`
Hyperparameters: search iteration N, iteration batch size K
for N iterations **do**
  `// Sample` K `reward code from LLM`
  $\texttt{R}_1, ..., \texttt{R}_k \sim \texttt{LLM}(l, M, \texttt{prompt})$
  `// Evaluate reward candidates`
  $s_1 = F(\texttt{R}_1), ..., s_K = F(\texttt{R}_K)$
  `// Reward reflection`
  $\texttt{prompt} := \texttt{prompt} : \texttt{Reflection}(R^n_{best}, s^n_{best})$,
  where $best = \arg\max_k s_1, ..., s_K$
  `// Update Eureka reward`
  $R_{\text{Eureka}}, s_{\text{Eureka}} = (R^n_{\text{best}}, s^n_{\text{best}}), \quad \text{if } s^n_{\text{best}} > s_{\text{Eureka}}$
Output: R_{Eureka}

the LLM's context window and does not leak simulation internals (so that we can expect the same prompt to generalize to new simulators), we have an automatic script to extract just the environment code snippets that expose and fully specify the environment state and action variables.

Given environment as context, EUREKA instructs the coding LLM to directly return executable Python code with only generic reward design and formatting tips, such as exposing individual components in the reward as a dictionary output (for reasons that will be apparent in Sec. 5.4). Remarkably, with only these minimal instructions, EUREKA can already zero-shot generate plausibly-looking rewards in diverse environments in its first attempts. An example EUREKA output is shown in Fig. 5.3. As seen, EUREKA adeptly composes over existing observation variables (e.g., `fingertip_pos`) in the provided environment code and produces a competent reward code – all without any environment-specific prompt engineering or reward templating. On the first try, however, the generated reward may not always be executable, and even if it is, it can be quite sub-optimal with respect to the task fitness metric F. While we can improve the prompt with task-specific formatting and reward design hints, doing so does not scale to new tasks and hinders the overall generality of our system. How can we effectively overcome the sub-optimality of single-sample reward generation?

Evolutionary Search

In this section, we will demonstrate how evolutionary search presents a natural solution that addresses the aforementioned execution error and sub-optimality

```
def compute_reward(object_rot, goal_rot, object_angvel, object_pos, fingertip_pos):
    # Rotation reward
    rot_diff = torch.abs(torch.sum(object_rot * goal_rot, dim=1) - 1) / 2
-   rotation_reward_temp = 20.0
+   rotation_reward_temp = 30.0                                          Changing hyperparameter
    rotation_reward = torch.exp(-rotation_reward_temp * rot_diff)

    # Distance reward
+   min_distance_temp = 10.0
    min_distance = torch.min(torch.norm(fingertip_pos - object_pos[:, None], dim=2), dim=1).values
-   distance_reward = min_distance
+   uncapped_distance_reward = torch.exp(-min_distance_temp * min_distance)
+   distance_reward = torch.clamp(uncapped_distance_reward, 0.0, 1.0)    Changing functional form

-   total_reward = rotation_reward + distance_reward
+   # Angular velocity penalty                                           Adding new component
+   angvel_norm = torch.norm(object_angvel, dim=1)
+   angvel_threshold = 0.5
+   angvel_penalty_temp = 5.0
+   angular_velocity_penalty = torch.where(angvel_norm > angvel_threshold,
+       torch.exp(-angvel_penalty_temp * (angvel_norm - angvel_threshold)), torch.zeros_like(angvel_norm))
+
+   total_reward = 0.5 * rotation_reward + 0.3 * distance_reward - 0.2 * angular_velocity_penalty

    reward_components = {
        "rotation_reward": rotation_reward,
        "distance_reward": distance_reward,
+       "angular_velocity_penalty": angular_velocity_penalty,
    }

    return total_reward, reward_components
```

Figure 5.3: EUREKA can zero-shot generate executable rewards and then flexibly improve them with many distinct types of free-form modification, such as (1) changing the hyperparameter of existing reward components, (2) changing the functional form of existing reward components, and (3) introducing new reward components.

challenges. In each iteration, EUREKA samples several independent outputs from the LLM (Line 5 in Alg. 1). Since the generations are i.i.d, the probability that *all* reward functions from an iteration are buggy *exponentially* decreases as the number of samples increases. We find that for all environments we consider, sampling just a modest number of samples (16) contains at least one executable reward code in the first iteration.

Given executable reward functions from an earlier iteration, EUREKA performs in-context *reward mutation*, proposing new improved reward functions from the best one in the previous iteration. Concretely, a new EUREKA iteration will take the best-performing reward from the previous iteration, its reward reflection (Sec. 5.4), and the mutation prompt as context and generate K more i.i.d reward outputs from the LLM; several illustrative reward modifications are visualized in Fig. 5.3. This iterative optimization continues until a specified number of iterations has been reached. Finally, we perform multiple random restarts to find better maxima; this is

a standard strategy in global optimization. In all our experiments, EUREKA conducts 5 independent runs per environment, and for each run, searches for 5 iterations with $K = 16$ samples per iteration.

Reward Reflection

In order to ground the in-context reward mutation, we must be able to put into words the quality of the generated rewards. We propose *reward reflection*, an automated feedback that summarizes the policy training dynamics in texts. Specifically, given that EUREKA reward functions are asked to expose their individual components in the reward program (e.g., `reward_components` in Fig. 5.3), reward reflection tracks the scalar values of all reward components and the task fitness function at intermediate policy checkpoints throughout training. For instance, consider the illustrative example in Fig. 5.2, where the snapshot values of `av_penalty` are provided as a list in the reward feedback.

This reward reflection procedure, though simple to construct, is important due to two reasons: (1) the lack of fine-grained reward improvement signal in the task fitness function, and (2) the algorithm-dependent nature of reward optimization (Booth et al., 2023). First, as we can query the task fitness function F on the resulting policies, a simple strategy is to just provide this numerical score as the reward evaluation. While serving as the holistic ground-truth metric, the task fitness function itself lacks in credit assignment, providing no useful information on *why* a reward function works or not. Second, whether a reward function is effective is influenced by the particular choice of RL algorithm, and the same reward may perform very differently even under the same optimizer given hyperparameter differences (Henderson et al., 2018; Agarwal et al., 2021). By providing detailed accounts on how well the RL algorithm optimizes individual reward components, reward reflection enables EUREKA to produce more intricate and targeted reward editing.

5.5 Experiments

We thoroughly evaluate EUREKA on a diverse suite of robot embodiments and tasks, testing its ability to generate reward functions, solve new tasks, and incorporate various forms of human input. We use GPT-4 (OpenAI, 2023), in particular the `gpt-4-0314` variant, as the backbone LLM for all LLM-based reward-design algorithms unless specified otherwise.

Environments. Our environments consist of 10 distinct robots and 29 tasks implemented using the IsaacGym simulator (Makoviychuk et al., 2021). First, we

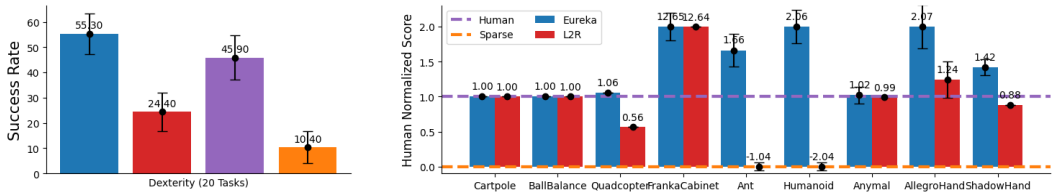


Figure 5.4: EUREKA outperforms Human and L2R across all tasks. In particular, EUREKA realizes much greater gains on high-dimensional dexterity environments.

include 9 original environments from IsaacGym (**Isaac**), covering a diverse set of robot morphologies from quadruped, bipedal, quadrotor, cobot arm, to dexterous hands. In addition to coverage over robot form factors, we ensure *depth* in our evaluation by including all 20 tasks from the Bidexterous Manipulation (**Dexterity**) benchmark (Chen et al., 2022c). Dexterity contains 20 complex bi-manual tasks that require a pair of Shadow Hands to solve a wide range of complex manipulation skills, ranging from object handover to rotating a cup by 180 degrees. For the task description input to EUREKA, we use the official description provided in the environment repository when possible. It is worth noting that both benchmarks are publicly released concurrently, or after the GPT-4 knowledge cut-off date (September 2021), so GPT-4 is unlikely to have accumulated extensive internet knowledge about these tasks, making them ideal testbeds for assessing EUREKA's reward generation capability compared to measurable human-engineered reward functions.

Baselines

L2R (Yu et al., 2023b) proposes a two-stage LLM-prompting solution to generate templated rewards. For an environment and task specified in natural language, a first LLM is asked to fill in a natural language template describing the agent's motion; then, a second LLM is asked to convert this "motion description" into code that calls a manually defined set of reward API primitives to write a reward program that sets their parameters. To make L2R competitive for our tasks, we define the motion description template to mimic the original L2R templates, and we construct the API reward primitives using the individual components of the original human rewards when possible. Note that this gives L2R an advantage as it has access to the original reward functions. Consistent with EUREKA, we conduct 5 independent L2R runs per environment, and for each run, we generate 16 reward samples.

Human. These are the original shaped reward functions used in our benchmark tasks. Designed by active reinforcement learning researchers, they reflect expert-level human reward engineering.

Sparse. These are identical to the fitness functions F that we use to evaluate the quality of the generated rewards. Like Human, these are also provided by the benchmark. On the dexterity tasks, they are uniformly binary indicator functions that measure task success; on Isaac tasks, they vary in functional forms depending on the nature of the task.

Training Details

Policy Learning. For each task, all final reward functions are optimized using the same RL algorithm with the same set of hyperparameters. Isaac and Dexterity share a well-tuned PPO implementation (Schulman et al., 2017; Makoviichuk et al., 2021), and we use this implementation and the task-specific PPO hyperparameters without any modification. Note that these task hyperparameters are tuned to make the official human-engineered rewards work well. For each final reward function obtained from each method, we run 5 independent PPO training runs and report the average of the maximum task metric values achieved from 10 policy checkpoints sampled at fixed intervals. In particular, the maximum is taken over the same number of checkpoints for each approach.

Reward Evaluation Metrics. For Isaac tasks, since the task metric F for each task varies in semantic meaning and scale, we report the **human normalized score** for EUREKA and L2R, $\frac{\texttt{Method}-\texttt{Sparse}}{|\texttt{Human}-\texttt{Sparse}|}$. This metric provides a holistic measure of how EUREKA rewards fare against human-expert rewards with respect to the ground-truth task metric. For Dexterity, since all tasks are evaluated using the binary success function, we directly report success rates.

Results

EUREKA outperforms human rewards. In Figure 5.4, we report the aggregate results on Dexterity and Isaac. Notably, EUREKA exceeds or performs on par to human level on *all* Isaac tasks and 15 out of 20 tasks on Dexterity. In contrast, L2R, while comparable on low-dimensional tasks (e.g., CartPole, BallBalance), lags significantly behind on high-dimensional tasks. Despite being provided access to some of the same reward components as Human, L2R still underperforms EUREKA after its initial iteration, when both methods have had the same number of reward queries. As expected, L2R's lack of expressivity severely limits its performance. In contrast, EUREKA generates free-form rewards from scratch without any domain-specific knowledge and performs substantially better. Furthermore, we ablate GPT-4 with GPT-3.5 and find EUREKA degrades in performance but still matches or exceeds

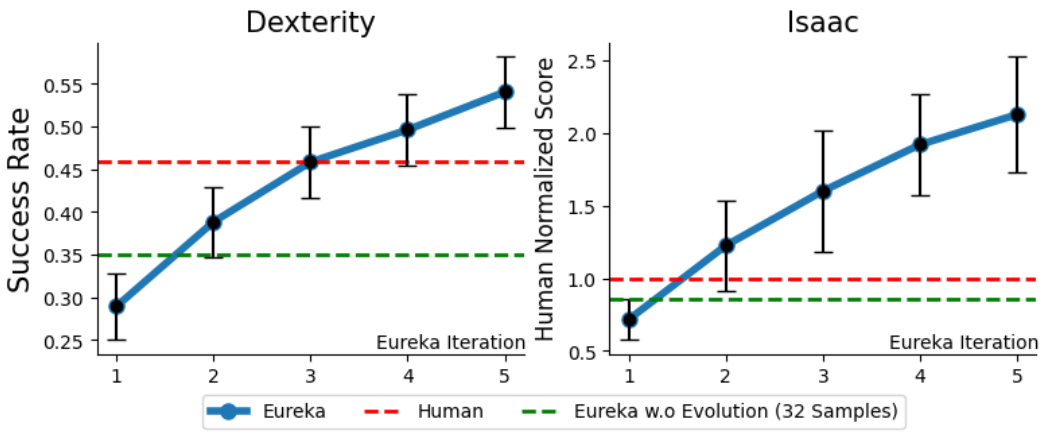


Figure 5.5: EUREKA progressively produces better rewards via in-context evolutionary reward search.

human-level on most Isaac tasks, indicating that its general principles can be readily applied to coding LLMs of varying qualities.

EUREKA consistently improves over time. In Fig. 5.5, we visualize the average performance of the cumulative best EUREKA rewards after each evolution iteration. Moreover, we study an ablation, **EUREKA w.o. Evolution (32 Samples)**, which performs only the initial reward generation step, sampling the same number of reward functions as two iterations in the original EUREKA. This ablation helps study, given a fixed number of reward function budget, whether it is more advantageous to perform the EUREKA evolution or simply sample more first-attempt rewards without iterative improvement. As seen, on both benchmarks, EUREKA rewards steadily improve and eventually surpass human rewards in performance despite sub-par initial performances. This consistent improvement also cannot be replaced by just sampling more in the first iteration as the ablation's performances are lower than EUREKA after 2 iterations on both benchmarks. Together, these results demonstrate that EUREKA's novel evolutionary optimization is indispensable for its final performance.

EUREKA generates novel rewards. We assess the novelty of EUREKA rewards by computing the *correlations* between EUREKA and human rewards on all the Isaac tasks. Then, we plot the correlations against the human normalized scores on a scatter-plot in Figure 5.6, where each point represents a single EUREKA reward on a single task. As shown, EUREKA mostly generates weakly correlated reward functions that outperform the human ones. In addition, by examining the average correlation by task, we observe that *the harder the task is, the less correlated the EUREKA rewards*. We hypothesize that human rewards are less likely to be near

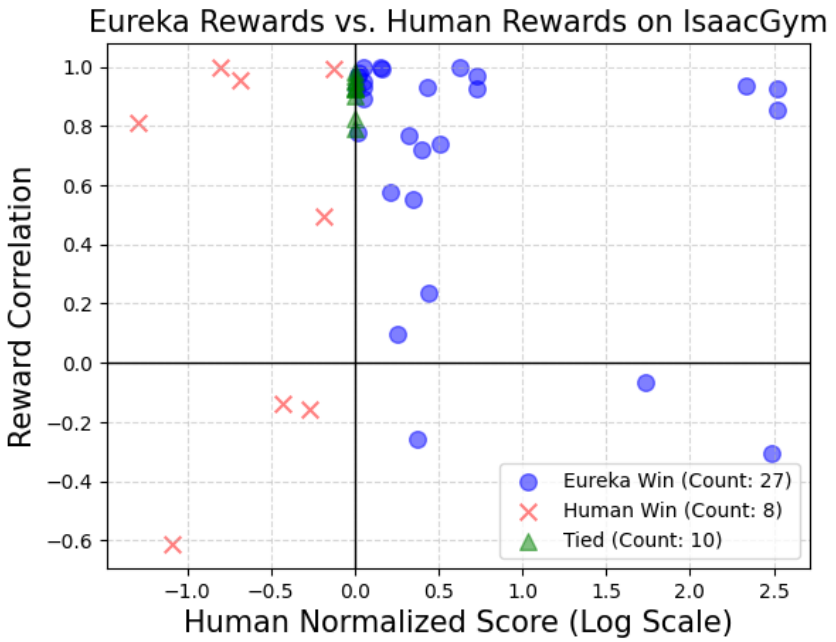


Figure 5.6: Eureka generates novel rewards.

optimal for difficult tasks, leaving more room for EUREKA rewards to be different and better. In a few cases, EUREKA rewards are even *negatively* correlated with human rewards but perform significantly better, demonstrating that EUREKA can *discover* novel reward design principles that may run counter to human intuition.

Reward reflection enables targeted improvement. To assess the importance of constructing reward reflection in the reward feedback, we evaluate an ablation, **EUREKA (No Reward Reflection)**, which reduces the reward feedback prompt to include only snapshot values of the task metric F. Averaged over all Isaac tasks, EUREKA without reward reflection reduces the average normalized score by 28.6%.

EUREKA with curriculum learning enables dexterous pen spinning. Finally, we investigate whether EUREKA can be used to solve a truly novel and challenging dexterous task. To this end, we propose pen spinning as a test bed. This task is highly dynamic and requires a Shadow Hand to continuously rotate a pen to achieve some pre-defined spinning patterns for as many cycles as possible; we implement this task on top of the original Shadow Hand environment in Isaac Gym without changes to any physics parameter, ensuring physical realism. We consider a *curriculum learning* (Bengio et al., 2009) approach to break down the task into manageable components that can be independently solved by EUREKA. Specifically, we first use EUREKA to generate a reward for the task of re-orienting the pen to random target

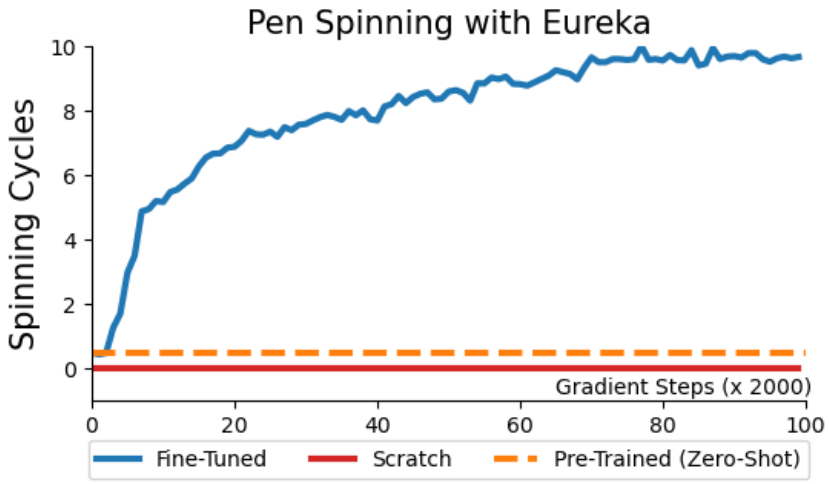


Figure 5.7: EUREKA can be flexibly combined with curriculum learning to acquire complex dexterous skills.

configurations and train a policy using the final EUREKA reward. Then, using this pre-trained policy (**Pre-Trained**), we fine-tune it using the same EUREKA reward to reach the sequence of pen-spinning configurations (**Fine-Tuned**). To demonstrate the importance of curriculum learning, we also directly train a policy from scratch on the target task using EUREKA reward without the first-stage pre-training (**Scratch**). The RL training curves are shown in Figure 5.7. Eureka fine-tuning quickly adapts the policy to successfully spin the pen for many cycles in a row; see project website for videos. In contrast, neither pre-trained or learning-from-scratch policies can complete even a single cycle of pen spinning. In addition, using this EUREKA fine-tuning approach, we have also trained pen spinning policies for a variety of different spinning configurations. These results demonstrate EUREKA's applicability to advanced policy learning approaches, which are often necessary for learning very complex skills

EUREKA From Human Feedback

In addition to automated reward design, EUREKA enables a new gradient-free in-context learning approach to RL from Human Feedback (RLHF) that can readily incorporate various types of human inputs to generate more performant and human-aligned reward functions.

EUREKA can improve and benefit from human reward functions. We study whether starting with a human reward function initialization, a common scenario in real-world RL applications, is advantageous for EUREKA. Importantly, incorporating

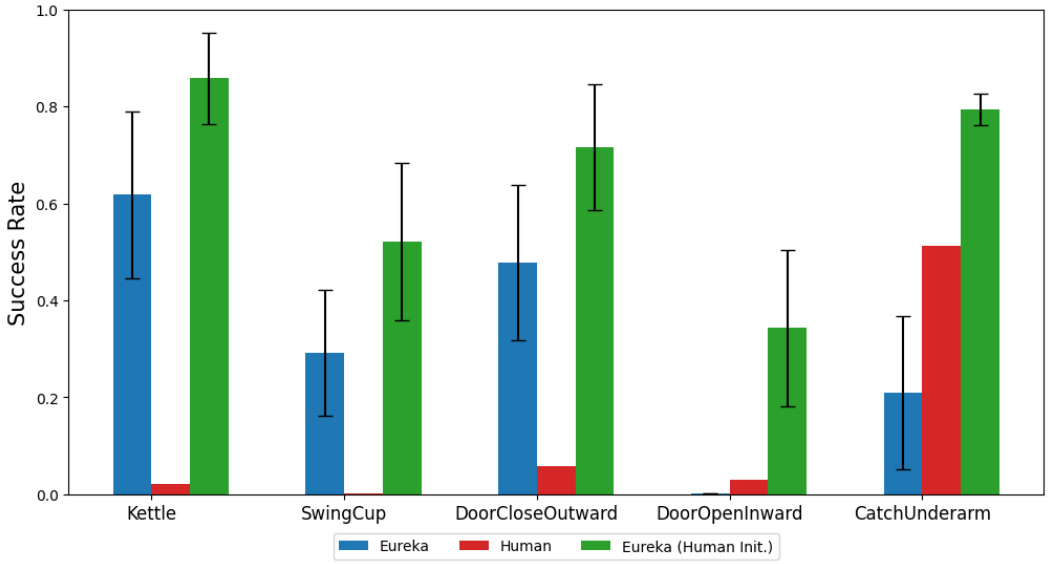
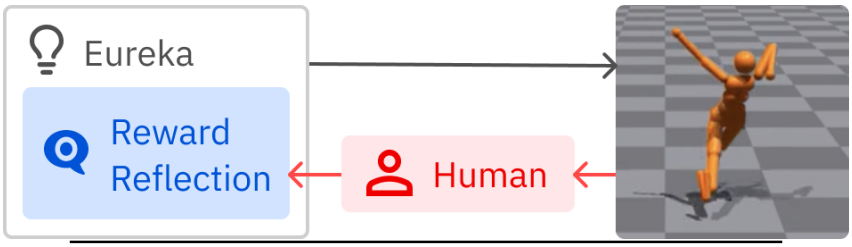


Figure 5.8: EUREKA effectively improves and benefits from human reward initialization.

human initialization requires no modification to EUREKA – we can simply substitute the raw human reward function as the output of the first EUREKA iteration. To investigate this, we select several tasks from Dexterity that differ in the relative performances between the original EUREKA and human rewards. The full results are shown in Figure 5.8.

As shown, regardless of the quality of the human rewards, EUREKA improves and benefits from human rewards as **EUREKA (Human Init.)** is uniformly better than both EUREKA and Human on all tasks. This suggests that EUREKA's in-context reward improvement capability is largely independent of the quality of the base reward. Furthermore, the fact that EUREKA can significantly improve over human rewards even when they are highly sub-optimal hints towards an interesting hypothesis: *human designers are generally knowledgeable about relevant state variables but are less proficient at designing rewards using them*. This makes intuitive sense as identifying relevant state variables that should be included in the reward function involves mostly common sense reasoning, but reward design requires specialized knowledge and experience in RL. Together, these results demonstrate EUREKA's *reward assistant* capability, perfectly complementing human designers' knowledge about useful state variables and making up for their less proficiency on how to design rewards using them.

Reward reflection via human feedback induces aligned behavior. So far, all EUREKA rewards are optimized against a fixed, black-box task fitness function F.



Method	Forward Velocity	Human Preference
EUREKA	**7.53**	5/20
EUREKA-HF	5.58	**15/20**

Figure 5.9: EUREKA can incorporate human reward reflection to modify rewards that induce safer and more human-aligned behavior.

This task metric, however, may not fully align with human intent. Moreover, in many open-ended tasks, F may not be available in the first place (Fan et al., 2022). In these challenging scenarios, we propose to augment EUREKA by having humans step in and put into words the reward reflection in terms of the desired behavior and correction. We investigate this capability in EUREKA by teaching a Humanoid agent how to run purely from textual reward reflection. Then, we conduct a user study asking 20 unfamiliar users to indicate their preferences between two policy rollout videos shown in random order, one trained with human reward reflection (**EUREKA-HF**) and the other one trained with the original best EUREKA reward. As shown in Fig. 5.9, despite running a bit slower, the EUREKA-HF agent is preferred by a large majority of our users. Qualitative, we indeed see that the EUREKA-HF agent acquires safer and more stable gait, as instructed by the human. See the project website for a comparison.

5.6 Conclusion

We have presented EUREKA, a universal reward design algorithm powered by coding large language models and in-context evolutionary search. Without any task-specific prompt engineering or human intervention, EUREKA achieves human-level reward generation on a wide range of robots and tasks. EUREKA's particular strength in learning dexterity solves dexterous pen spinning for the first time with a curriculum learning approach. Finally, EUREKA enables a gradient-free approach to reinforcement learning from human feedback that readily incorporates human reward initialization and textual feedback to better steer its reward generation. The versatility and substantial performance gains of EUREKA suggest that the simple principle of combining large language models with evolutionary algorithms are a general and

scalable approach to reward design, an insight that may be generally applicable to difficult, open-ended search problems.

Chapter 6

VISION LANGUAGE ACTION MODELS FOR ROBOT MANIPULATION IN SIMULATION

This chapter is based on the paper:

Jiang, Yunfan, Agrim Gupta*, Zichen Zhang*, Guanzhi Wang*, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan (July 2023). VIMA: Robot Manipulation With Multimodal Prompts. In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 14975–15022. URL: https://proceedings.mlr.press/v202/jiang23b.html.

6.1 Introduction

Transformers have given rise to remarkable multi-task consolidation across many AI domains. For example, users can describe a task using natural language prompt to GPT-3 (Brown et al., 2020), allowing the same model to perform question answering, machine translation, text summarization, etc. Prompt-based learning provides an accessible and flexible interface to communicate a natural language understanding task to a general-purpose model.

We envision that a generalist robot agent should have a similarly intuitive and expressive interface for task specification. What does such an interface for robot learning look like? As a motivating example, consider a personal robot tasked with household activities. We can ask the robot to bring us a cup of water by a simple natural language instruction. If we require more specificity, we can instead instruct the robot to "bring me <image of the cup>." For tasks requiring new skills, the robot should be able to adapt preferably from a few video demonstrations (Duan et al., 2017). Tasks that need interaction with unfamiliar objects can be easily explained via a few image examples for *novel concept grounding* (Hermann et al., 2017). Finally, to ensure safe deployment, we can further specify visual constraints like "do not enter <image> room."

To enable a single agent with all these capabilities, we make three key contributions in this work: 1) a novel **multimodal prompting formulation** that converts a wide spectrum of robot manipulation tasks into one sequence modeling problem; 2) a new

robot agent model capable of multi-tasking and zero-shot generalization; and 3) a **large-scale benchmark** with diverse tasks to systematically evaluate the scalability and generalization of our agents.

We start with the observation that many robot manipulation tasks can be formulated by **multimodal prompts that interleave language and images or video frames**. For example, Rearrangement (Batra et al., 2020), a type of *Visual Goal*, can be formulated as "Please rearrange objects to match this `{scene_image}`"; *Novel Concept Grounding* looks like "This is a dax $\texttt{\{new\_object\}}_1$ and this is a blicket $\texttt{\{new\_object\}}_2$. Put two metal dax on the marble blicket."; *Few-shot Imitation* can embed video snippet in the prompt "Follow this motion trajectory for the wooden cube: $\{\texttt{frame}_1\}, \{\texttt{frame}_2\}, \{\texttt{frame}_3\}, \{\texttt{frame}_4\}$"; and expressing *Visual Constraint* is as simple as adding the clause "without touching `{safety_boundary}`."

Multimodal prompts not only have more expressive power than individual modalities, but also enable a **uniform sequence IO interface** for training generalist robot agents. Previously, different robot manipulation tasks require distinct policy architectures, objective functions, data pipelines, and training procedures (Aceituno et al., 2021; Stengel-Eskin et al., 2022; Lynch et al., 2021), leading to siloed robot systems that cannot be easily combined for a rich set of use cases. Instead, our multimodal prompt interface allows us to harness the latest advances in large transformer models (Lin et al., 2021a; Tay et al., 2020; Khan et al., 2021) for developing scalable multi-task robot learners.

To this end, we design a novel **Vi**suo**M**otor **A**ttention model (VIMA, reads "*v-eye-ma*"). The architecture follows the encoder-decoder transformer design proven to be effective and scalable in NLP (Raffel et al., 2020). VIMA encodes an input sequence of interleaving textual and visual prompt tokens with a pretrained language model (Tsimpoukelli et al., 2021), and decodes robot control actions autoregressively for each environment interaction step. The transformer decoder is conditioned on the prompt via cross-attention layers that alternate with the usual causal self-attention. Instead of operating on raw pixels, VIMA adopts an object-centric approach. We parse all images in the prompt or observation into objects by off-the-shelf detectors (He et al., 2017), and flatten them into sequences of object tokens. All these design choices combined deliver a conceptually simple architecture with strong model and data scaling properties.

To systematically evaluate our proposed algorithm, we introduce a new benchmark (VIMA-Bench) built on the Ravens simulator (Zeng et al., 2020; Shridhar et al.,

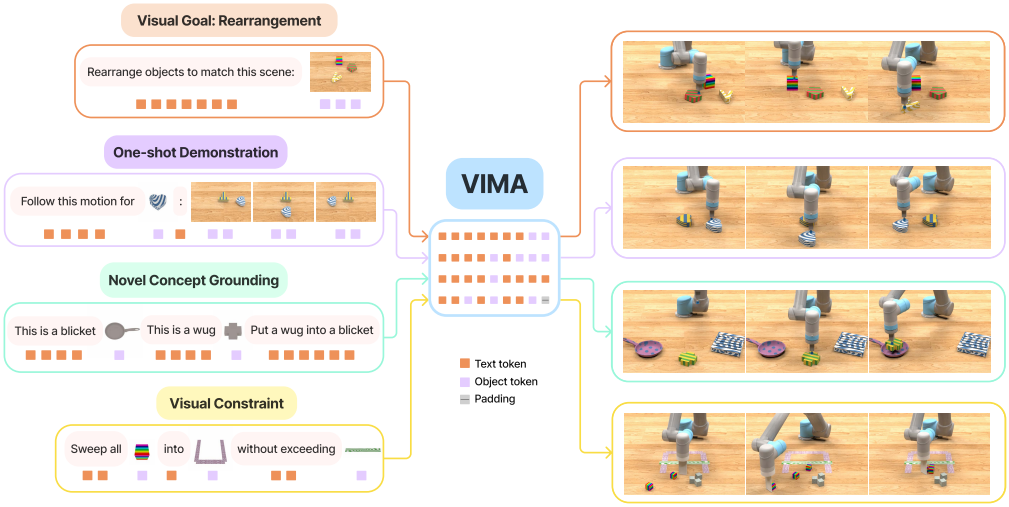


Figure 6.1: Multimodal prompts for task specification. We observe that many robot manipulation tasks can be expressed as *multimodal prompts* that interleave language and image/video frames. We propose VIMA, an embodied agent model capable of processing mulitimodal prompts (left) and controlling a robot arm to solve the task (right).

2021). We provide 17 representative meta-tasks with multimodal prompt templates, which can be procedurally instantiated into thousands of individual tasks by various combinations of textures and tabletop objects. VIMA-BENCH establishes a 4-level protocol to evaluate progressively stronger generalization capabilities, from randomized object placement to novel tasks altogether (Fig. 6.2). To demonstrate the scalability of VIMA, we train a spectrum of 7 models ranging from 2M to 200M parameters. Our approach outperforms strong prior SOTA methods such as Gato (Reed et al., 2022), Decision Transformer (Chen et al., 2021b), and Flamingo (Alayrac et al., 2022) across all 4 levels of zero-shot generalization and all model capacities, sometimes by a large margin (up to $2.9\times$ task success rate given the same amount of training data, and $2.7\times$ better even with $10\times$ less data). We open-source the simulation suite, training dataset, algorithm implementation, and pretrained model checkpoints to ensure reproducibility and facilitate future works from the community.

6.2 Related Work

Multi-task Learning by Sequence Modeling. Transformers have enabled task unification across many AI domains (Raffel et al., 2020; Radford et al., 2019; Brown et al., 2020; Chen et al., 2022a; Chen et al., 2022b; Lu et al., 2022; Wang et al., 2022b;

Alayrac et al., 2022; Reed et al., 2022). For example, in **NLP**, T5 (Raffel et al., 2020) unifies all language problems into the same text-to-text format. GPT-3 (Brown et al., 2020), PaLM (Chowdhery et al., 2022), and Megatron-Turing NLG (Shoeybi et al., 2019) demonstrate emergent behaviours of intuitive task specifications by zero-shot prompting without any finetuning. In **computer vision**, Florence (Yuan et al., 2021), BiT (Kolesnikov et al., 2020), and MuST (Ghiasi et al., 2021) pre-train a shared backbone model at scale for general visual representations and transfer it across downstream vision tasks. Pix2Seq (Chen et al., 2022b) casts many vision problems into a unified sequence format. In **multimodal learning**, Flamingo (Alayrac et al., 2022), Frozen (Tsimpoukelli et al., 2021), and VL-T5 (Cho et al., 2021) design a universal API that ingests an interleaving sequence of images and text and generates free-form text. Gato (Reed et al., 2022) and HighMMT (Liang et al., 2022) are massively multi-task models across NLP, vision, and embodied agents. Our work is most similar in spirit to Gato, but we focus primarily on enabling an intuitive, multimodal prompting interface for a generalist robot agent.

Foundation Models for Embodied Agents. Foundation models (Bommasani et al., 2021; Brown et al., 2020; Raffel et al., 2020; Ramesh et al., 2022; Wei et al., 2022a) have demonstrated strong emergent properties like zero-shot prompting and complex reasoning. There are many ongoing efforts to replicate this success for embodied agents, focusing on 3 aspects: 1) **Transformer agent architecture**: Decision Transformer (Chen et al., 2021b; Janner et al., 2021; Zheng et al., 2022; Xu et al., 2022) and Gato (Reed et al., 2022) leverage the powerful self-attention models for sequential decision making. CLIPort (Shridhar et al., 2021) and Perceiver-Actor (Shridhar et al., 2022) apply large transformers to robot manipulation tasks. Less is More (Wang et al., 2021) generates navigation instructions from a mixture of visual landmarks and text. 2) **Pre-training for better representations**: MVP (Xiao et al., 2022), R3M (Nair et al., 2022), and Parisi et al. (2022) pre-train general visual representations for robotic perception. Li et al. (2022a) and Reid et al. (2022) finetune from LLM checkpoints to accelerate policy learning. MineDojo (Fan et al., 2022) and Ego4D (Grauman et al., 2022) provide large-scale multimodal databases to facilitate scalable policy training. 3) **Large language models for robot learning**: SayCan (Ahn et al., 2022) leverages the 500B PaLM (Chowdhery et al., 2022) for zero-shot concept grounding. Socratic Models (Zeng et al., 2022) composes multiple vision and language foundation models (VLMs) for multimodal reasoning in videos. Huang et al. (2022a), Inner Monologue (Huang et al., 2022b) and LM-Nav (Shah et al., 2022) successfully apply LLMs

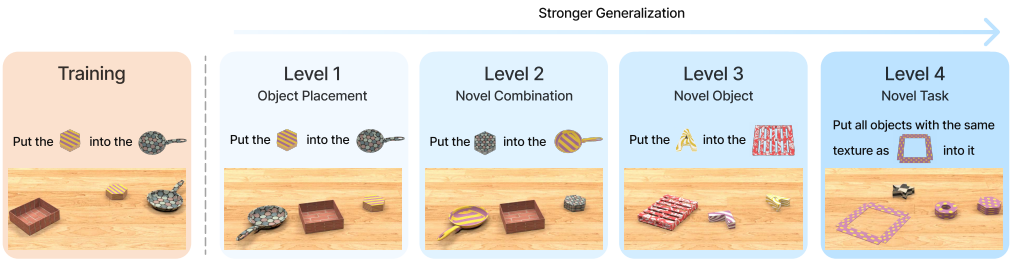


Figure 6.2: Evaluation protocol in VIMA-BENCH. We design 4 levels of evaluation settings to measure the zero-shot generalization capability of an agent systematically. Each level deviates more from the training distribution, and thus is strictly more challenging than the previous level.

to long-horizon robot planning. VIMA differs from these works in our novel multimodal prompting formulation, which existing LLMs and VLMs do not easily support.

Robot Manipulation and Benchmarks. There are a wide range of robot manipulation tasks that require different skills and task specification formats, such as instruction following (Stepputtis et al., 2020; Shridhar et al., 2021; Lynch et al., 2021), one-shot imitation (Finn et al., 2017; Dasari et al., 2020; Duan et al., 2017), rearrangement (Batra et al., 2020; Weihs et al., 2021; Szot et al., 2021), constraint satisfaction (Brunke et al., 2021; Srinivasan et al., 2020; Thananjeyan et al., 2021), and reasoning (Shridhar et al., 2020; Gupta et al., 2019; Ahmed et al., 2021; Toyer et al., 2020; Lim et al., 2021). Multiple physics simulation benchmarks are introduced to study the above tasks. For example, iGibson (Shen et al., 2020; Li et al., 2021; Srivastava et al., 2021) simulates interactive household scenarios. Ravens (Zeng et al., 2020) and Robosuite (Zhu et al., 2020c; Fan et al., 2021) design various tabletop manipulation tasks with realistic robot arms. Our VIMA-BENCH is the first robot learning benchmark to support multimodal-prompted tasks. We also standardize the evaluation protocol to systematically measure an agent's generalization capabilities.

6.3 Multimodal Prompts for Task Specification

A central and open problem in robot learning is task specification (Agrawal, 2022). Our key insight is that various task specification paradigms (such as goal conditioning, video demonstration, natural language instruction) can all be instantiated as multimodal prompts (Fig. 6.1). Concretely, a multimodal prompt $\mathcal{P}$ of length l is defined as an ordered sequence of arbitrarily interleaved texts and images $\mathcal{P} := \Big[x_0, x_1, \ldots, x_l\Big]$, where each element $x_i \in \{\text{text}, \text{image}\}$.

Task Suite. The flexibility afforded by multimodal prompts allows us to specify and build models for a huge variety of task specification formats. Here we consider the following six task categories:

1. **Simple object manipulation**: simple tasks like "put <object> into <container>," where each image in the prompt corresponds to a single object;
2. **Visual goal reaching**: manipulating objects to reach a goal configuration, *e.g.*, *Rearrangement* (Batra et al., 2020);
3. **Novel concept grounding**: the prompt contains unfamiliar words like "dax" and "blicket," which are explained by in-prompt images and then immediately used in an instruction. This tests the agent's ability to rapidly internalize new concepts;
4. **One-shot video imitation**: watching a video demonstration and learning to reproduce the same motion trajectory for a particular object;
5. **Visual constraint satisfaction**: the robot must manipulate the objects carefully and avoid violating the (safety) constraints;
6. **Visual reasoning**: tasks that require reasoning skills, such as appearance matching "move all objects with same textures as <object> into a container," and visual memory, "put <object> in container and then restore to their original position."

Note that these six categories are not mutually exclusive. For example, a task may introduce a previously unseen verb (*Novel Concept*) by showing a video demonstration, or combine goal reaching with visual reasoning.

6.4 Vima-Bench: Benchmark for Multimodal Robot Learning

Simulation Environment. Existing benchmarks are generally geared toward a particular task specification. To our knowledge, there is no benchmark that provides a rich suite of multimodal tasks and a comprehensive testbed for targeted probing of agent capabilities. To this end, we introduce a new benchmark suite for multimodal robot learning that we call VIMA-Bench. We built our benchmark by extending the Ravens robot simulator (Zeng et al., 2020; Shridhar et al., 2021). VIMA-Bench supports extensible collections of objects and textures to compose multimodal prompts and procedurally generate a large number of tasks. Specifically, we provide

17 meta-tasks with multimodal prompt templates, which can be instantiated into 1000s of individual tasks. Each meta-task belongs to one or more of 6 task specification methods mentioned above. VIMA-BENCH can generate large quantities of imitation learning data via scripted oracle agents.

Observation and Actions. The observation space of our simulator includes RGB images rendered from both frontal view and top-down view. Groundtruth object segmentations and bounding boxes are also provided for training object-centric models (Sec. 6.5). We inherit the high-level action space from Zeng et al. (2020), which consists of primitive motor skills like "pick and place" and "wipe." These are parameterized by poses of the end effector. Our simulator also features scripted oracle programs that can generate expert demonstrations by using privileged simulator state information, such as the precise location of all objects, and the groundtruth interpretation of the multimodal instruction.

Training Dataset. We leverage the pre-programmed oracles to generate a large offline dataset of expert trajectories for imitation learning. Our dataset includes 50K trajectories per meta-task, and 650K successful trajectories in total. We hold out a subset of object geometries and textures for evaluation, and designate 4 out of 17 meta-tasks as a testbed for zero-shot generalization.

Evaluating Zero-Shot Generalization. Each task in VIMA-BENCH has a binary success criterion and does not provide partial reward signals. During test time, we execute the agent policies in the physics simulator for multiple episodes to compute a success rate in percentage. The average success rate over all evaluated meta-tasks will be the final reported metric.

We design a 4-level evaluation protocol (Fig. 6.2) to systematically probe the generalization capabilities of learned agents. Each level deviates more from the training distribution, and is thus strictly harder than the previous one:

1. **Placement generalization**: all prompts are seen verbatim during training, but only the placement of objects on the tabletop is randomized at testing.
2. **Combinatorial generalization**: all materials (adjectives) and 3D objects (nouns) are seen during training, but new combinations of them appear in testing.
3. **Novel object generalization**: test prompts and the simulated workspace include novel adjectives and objects.

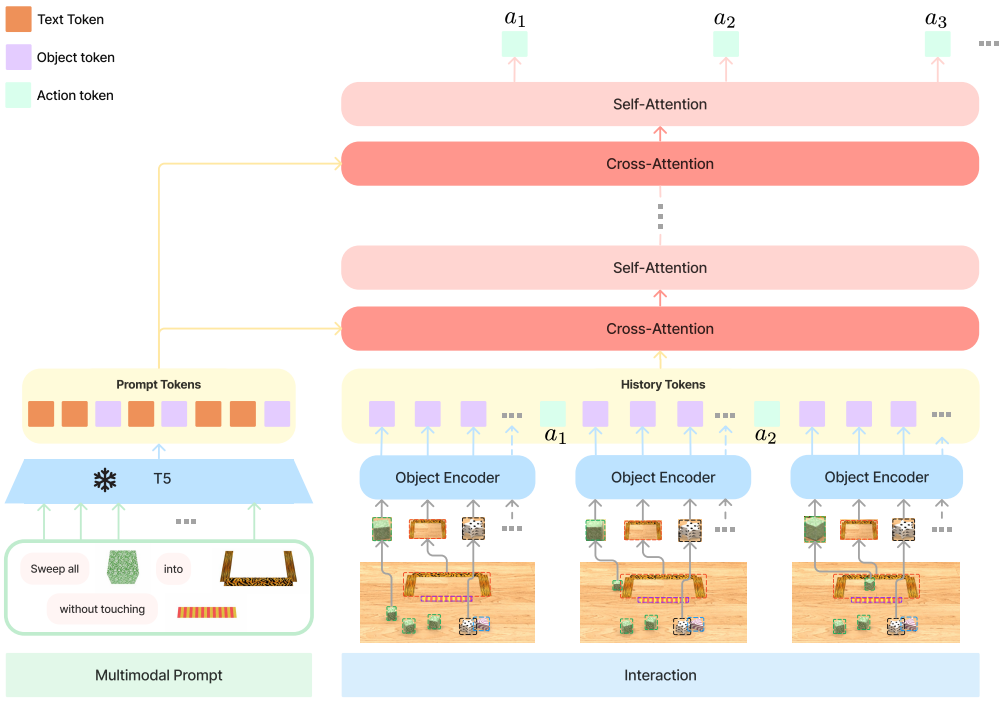


Figure 6.3: VIMA. We encode the multimodal prompts with a pre-trained T5 model, and condition the robot controller on the prompt through cross-attention layers. The controller is a causal transformer decoder consisting of alternating self and cross attention layers that predicts motor commands conditioned on prompts and interaction history.

4. **Novel task generalization**: new meta-tasks with novel prompt templates at test time.

6.5 VIMA: Visuomotor Attention Model

Our goal is to build a robot agent which can perform any task specified by multimodal prompts. To learn an effective multi-task robot policy, we propose VIMA, a minimalistic multi-task encoder-decoder architecture with object-centric design (Fig. 6.3). Concretely, we learn a robot policy $\pi(a_t|\mathcal{P}, \mathcal{H})$, where $\mathcal{H} := \left[o_1, a_1, o_2, a_2, \ldots, o_t\right]$ denotes the past interaction history, and $o_t \in \mathcal{O}, a_t \in \mathcal{A}$ are observations and actions at each interaction step. We encode the prompt via a *frozen* pre-trained langauge model and decode the robot motor commands conditioned on the encoded prompt via cross-attention layers.

Tokenization. There are 3 formats of raw input in the prompt — text, image of a single object, and image of a full tabletop scene (*e.g.*, for *Rearrangement* or imitation from video frames). For **text inputs**, we use pre-trained T5 tokenizer and

word embedding to obtain word tokens. For **images of full scenes**, we first extract individual objects using off-the-shelf Mask R-CNN (He et al., 2017). Each object is represented as a bounding box and a cropped image. We then compute object tokens by encoding them with a bounding box encoder and a ViT, respectively. For **images of single objects**, we obtain tokens in the same way except with a dummy bounding box. We then follow Tsimpoukelli et al. (2021) to produce prompt encoding by passing the resulted token sequence to a pre-trained T5 encoder model. Our positional embedding is learnable and absolute.

Robot Controller. A challenging aspect of designing multi-modal multi-task policy is to select a suitable conditioning mechanism. In our schema (Fig. 6.3), the robot controller (decoder) is conditioned on the prompt sequence $\mathcal{P}$ by a series of cross-attention layers between $\mathcal{P}$ and the trajectory history sequence $\mathcal{H}$. Each cross-attention layer generates an output sequence $\mathcal{H}' = \text{softmax}\left(\frac{Q_{\mathcal{H}} K_{\mathcal{P}}^{\intercal}}{\sqrt{d}}\right) V_{\mathcal{P}}$, where d is the embedding dimension. This design choice enjoys three advantages, including 1) strengthened connection to prompt, 2) intact and deep flow of the original prompt tokens, and 3) better computational efficiency. VIMA decoder consists of L alternating cross-attention and self-attention layers. Finally, for ease of learning and optimization, we follow common practice (Baker et al., 2022) to map predicted action tokens to discretized poses of the robot arm.

Training. We follow the standard behavioral cloning to train our models by minimizing the negative log-likelihood of predicted actions. Concretely, for a trajectory with T steps, we minimize $\min_\theta \sum_{t=1}^{T} -\log \pi_\theta(a_t|\mathcal{P}, \mathcal{H})$. The entire training is conducted on an offline dataset with no access to the physics simulator. To make VIMA robust to detection inaccuracies and failures, we apply *object augmentation* by randomly injecting *false-positive* detection outputs. After training, we select model checkpoints for evaluation based on the aggregated accuracy on a held-out validation set. The evaluation involves interacting with the physics simulator. We follow the best practices to train Transformer models using the AdamW optimizer (Loshchilov et al., 2019), learning rate warm-up, cosine annealing (Loshchilov et al., 2016), etc.

6.6 Experiments

In this section, we aim to answer three main questions:

1. How does VIMA compare with prior SOTA transformer-based agents on a diverse collection of multimodal-prompted tasks?

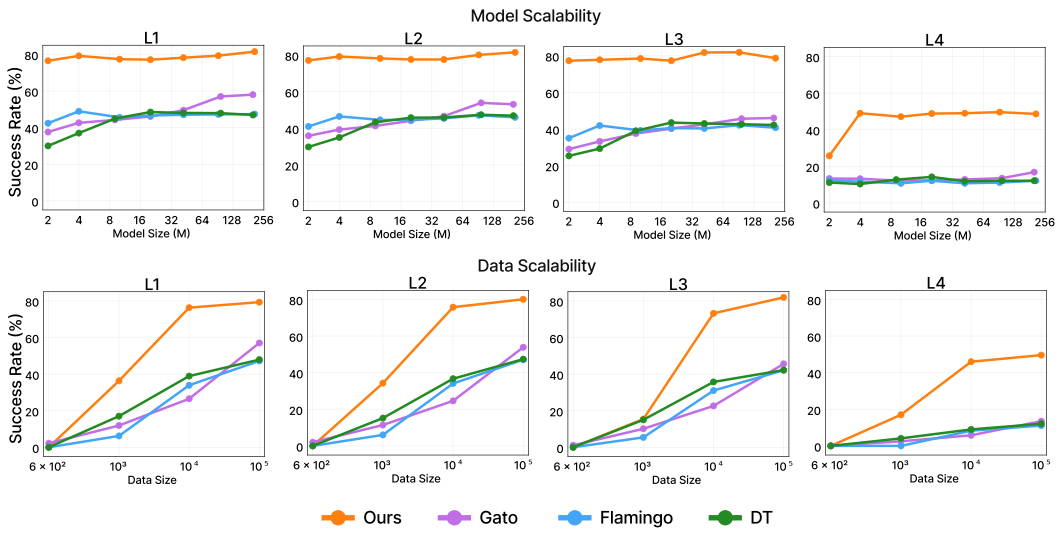


Figure 6.4: Scaling model and data. *Top:* We compare performance of different methods with model sizes ranging from 2M to 200M parameters. Across all model sizes and generalization levels VIMA outperforms prior works. *Bottom:* For a fixed model size of 92M parameters we compare the effect of imitation learning dataset size of 0.1%, 1%, 10%, and full imitation data. VIMA is extremely sample efficient and can achieve performance comparable to other methods with $10\times$ less data.

2. What are the **scaling properties** of our approach in model capacity and data size?

3. How do different visual tokenizers, prompt conditioning, and prompt encoding affect decision making?

Baselines

Gato (Reed et al., 2022) introduces a decoder-only model that solves tasks from multiple domains where tasks are specified by prompting the model with the observation and action subsequence. For fair comparison, we provide the same conditioning as VIMA, *i.e.*, our multimodal embedded prompt. Input images are divided into patches and encoded by a ViT (Dosovitskiy et al., 2021) model to produce observation tokens.

Flamingo (Alayrac et al., 2022) is a vision-language model that learns to generate textual completion in response to multimodal prompts. It embeds a variable number of prompt images into a fixed number of tokens via the Perceiver Resampler module, and conditions the language decoder on the encoded prompt by cross-attention. Flamingo does not work with embodied agents out of the box. We adapt it to support decision-masking by replacing the output layer with robot action heads.

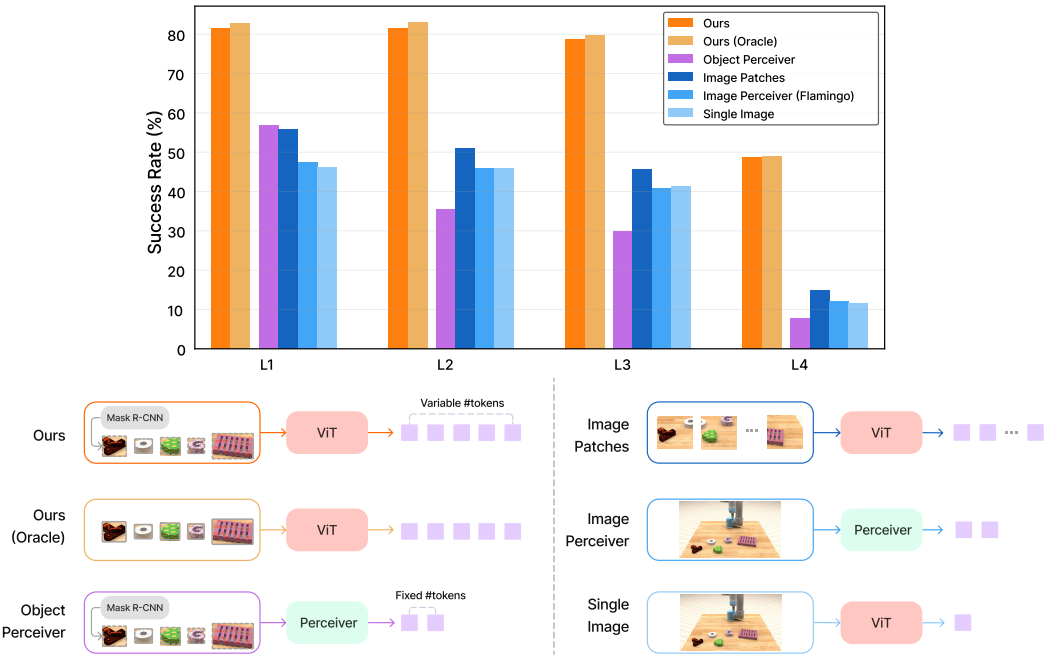


Figure 6.5: Ablation on visual tokenizers. We compare the performance of VIMA-200M model across different visual tokenizers. Our proposed object tokens outperform all methods that learn directly from raw pixels, and *Object Perceiver* that downsamples the object sequence to a fixed number of tokens.

Decision Transformer (DT) (Chen et al., 2021b; Janner et al., 2021) is among the first works to reinterpret the RL problem as transformer sequence modeling. In visual RL domains like Atari games, DT is prompted on the desired reward value and outputs actions autoregressively given the RGB observation embedding. We replace DT's initial reward prompt with our multimodal task prompt embeddings, and remove all subsequent reward tokens.

Evaluation Results

We compare VIMA against other SOTA methods on the four levels of generalization provided in our benchmark for different model and training dataset sizes.

Model scaling. We train all methods for a spectrum of model capacities from 2M to 200M parameters, evenly spaced on the log scale. The encoder size is kept constant (pretrained T5-Base) for all methods and excluded from the parameter count. Across *all* levels of zero-shot generalization, we find that VIMA strongly outperforms prior work. Although models like Gato and Flamingo show improved performance with bigger model sizes, VIMA consistently achieves superior performance over *all* model sizes. We note that this can only be achieved with *both* cross-attention and object token

sequence representation — altering any component will degrade the performance significantly, especially in the low model capacity regime (ablations in Sec. 6.6).

Data scaling. Next we investigate how different methods scale with varying dataset sizes. We compare model performance at 0.1%, 1%, 10% and full imitation learning dataset provided in VIMA-BENCH (Fig. 6.4). VIMA is extremely sample efficient and with just 1% of the data can achieve performance similar to baseline methods trained with $10\times$ more data for L1 and L2 levels of generalization. In fact, for L4 we find that with just 1% of training data, VIMA already outperforms prior work trained with *entire* dataset. Finally, across all levels with just 10% of the data, VIMA can outperform prior work trained with the full dataset by a significant margin. We hypothesize that the data efficiency can be attributed to VIMA's object-centric representation, which is less prone to overfitting than learning directly from pixels in the low-data regime. This is consistent with findings from Sax et al. (2018), which demonstrates that embodied agents conditioned on mid-level visual representations tend to be significantly more sample-efficient than end-to-end control from raw pixels.

Progressive Generalization. Finally, we compare the relative degradation in performance as we test the models on progressively challenging zero-shot evaluation levels without further finetuning (Fig. 6.6). Our method exhibits a minimal performance regression, especially between $L1 \rightarrow L2$ and $L1 \rightarrow L3$. In contrast, other methods can degrade as much as 20%, particularly in the more difficult generalization testing scenarios. Although all methods degrade significantly when evaluated on $L4$ (*Novel Tasks*), the drop in performance for VIMA is only *half* as severe as all other baselines. This results suggest that VIMA has developed more generalizable policy and robust representations than the competing approaches.

Ablation Studies

Through extensive experiments, we ablate different design choices in VIMA and study their impact on robot decision making. We focus on the following 3 aspects: visual tokenization, prompt encoding, and prompt conditioning variants.

Visual tokenization. As explained in Sec. 6.5, VIMA processes the prompt and observation images into a variable number of object tokens with an off-the-shelf Mask R-CNN implementation. How important is this particular choice of visual tokenizer? We study 5 different variants and empirically evaluate their 4 levels of

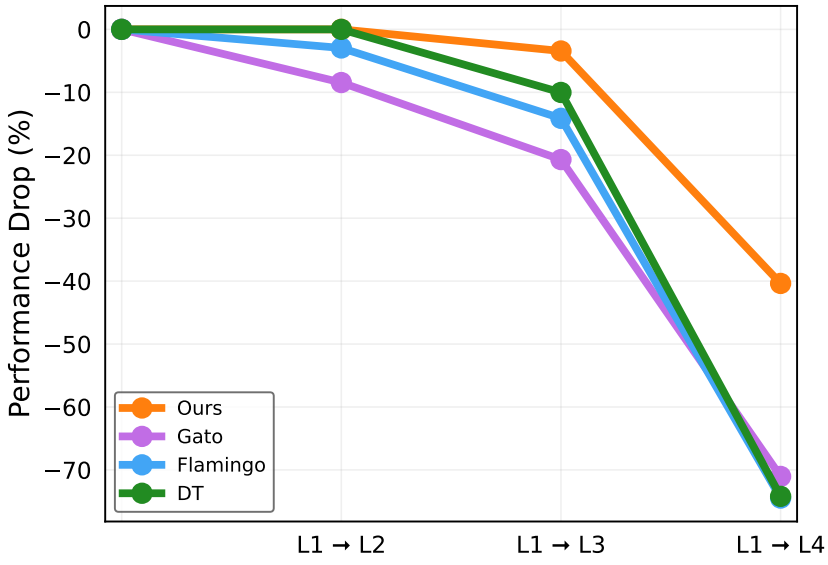


Figure 6.6: VIMA incurs much less performance drop than baselines as we evaluate on progressively harder zero-shot generalization.

generalization performance on VIMA-BENCH. (1) **Ours (Oracle)**: instead of using Mask R-CNN, we directly read out the groundtruth bounding box from the simulator. In other words, we use a perfect object detector to estimate the upper bound on the performance of this study; (2) **Object Perceiver**: we apply a Perceiver module (Jaegle et al., 2021b; Jaegle et al., 2021a) to convert the variable number of objects detected in each frame to a *fixed* number of tokens. Perceiver is more computationally efficient because it reduces the average sequence length; (3) **Image Perceiver**: the same architecture as the *Perceiver Resampler* in Flamingo, which converts an image to a small, fixed number of tokens; (4) **Image patches**: following Gato, we divide an RGB frame into square patches, and extract ViT embedding tokens. The number of patches is more than the output of Image Perceiver; (5) **Single image**: Decision Transformer's tokenizer, which encodes one image into a single token.

Fig. 6.5 shows the ablation results. We highlight a few findings. First, we note that our Mask R-CNN detection pipeline **incurs a minimal performance loss** compared to the oracle bounding boxes, thanks to the object augmentation (Sec. 6.5) that boosts robustness during training. Second, tokenizing from raw pixels (Image Perceiver, patches, or single embedding) consistently underperforms our object-centric format. We hypothesize that these tokenizers have to allocate extra internal capacity to parse the objects from low-level pixels, which likely impedes learning. Sax et al. (2018) echoes our finding that using mid-level vision can greatly improve agent generalization

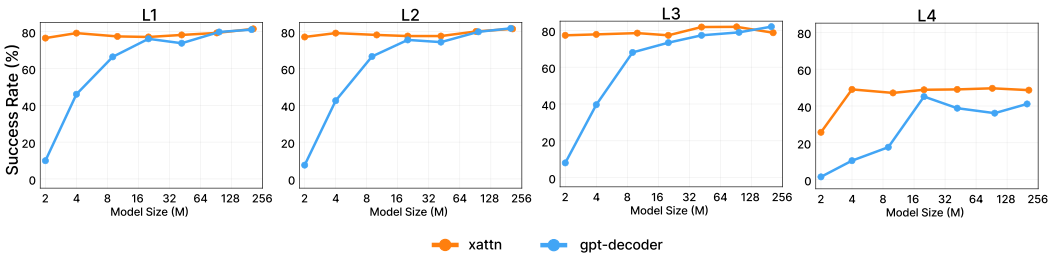


Figure 6.7: Ablation: prompt conditioning. We compare our method (*xattn*: cross-attention prompt conditioning) with a vanilla transformer decoder (*gpt-decoder*) across different model sizes. Cross-attention is especially helpful in low-parameter regime and for harder generalization tasks.

compared to an end-to-end pipeline. Third, even though *Ours* and *Object Perceiver* both use the same object bounding box inputs, the latter is significantly worse in decision making. We conclude that it is important to pass the **variable sequence of objects directly** to the robot controller rather than downsampling to a fixed number of tokens.

Prompt Conditioning. VIMA conditions the robot controller (decoder) on the encoded prompt by cross-attention. A simple alternative is to concatenate the prompt $\mathcal{P}$ and interaction history $\mathcal{H}$ into one big sequence, and then apply a decoder-only transformer like GPT (Radford et al., 2018) to predict actions. In this ablation, we keep the object tokenizer constant, and only switch the conditioning mechanism to causal sequence modeling. Fig. 6.7 shows the comparison of VIMA (`xattn`) and the `gpt-decoder` variant across 4 generalization levels. While GPT achieves comparable performance in larger models, cross-attention still dominates in the small-capacity range and generalizes better in the most challenging L4 (*Novel Task*) setting. Our hypothesis is that cross-attention helps the controller stay better focused on the prompt instruction at each interaction step. This bears resemblance to the empirical results in Sanh et al. (2021) and Wang et al. (2022a), which show that well-tuned encoder-decoder architectures can outperform GPT-3 in zero-shot generalization.

Prompt Encoding. We vary the size of the pre-trained T5 encoder (Raffel et al., 2020) to study the effect of prompt encoding. We experiment with three T5 model capacities: `t5-small` (30M), `t5-base` (111M), to `t5-large` (368M). For all T5 variants, we fine-tune the last two layers and freeze all other layers. We find no significant difference among the variants, thus we set `t5-base` as default for all our models.

6.7 Conclusion

Similar to GPT-3, a generalist robot agent should have an intuitive and expressive interface for human users to convey their intent. In this work, we introduce a novel *multimodal* prompting formulation that converts diverse robot manipulation tasks into a uniform sequence modeling problem. We propose VIMA, a conceptually simple transformer-based agent capable of solving tasks like visual goal, one-shot video imitation, and novel concept grounding with a single model. VIMA exhibits superior model and data scaling properties, and provides a strong starting point for future work.

Chapter 7

VISION LANGUAGE ACTION MODELS FOR HUMANOID ROBOTS IN THE REAL WORLD

This chapter is based on the paper:

NVIDIA, Guanzhi Wang, et al. (2025). Gr00t N1: An Open Foundation Model for Generalist Humanoid Robots. In: *arXiv Preprint arXiv:2503.14734*. URL: https://arxiv.org/abs/2503.14734.

7.1 Introduction

Creating autonomous robots to perform everyday tasks in the human world has long been a fascinating goal and, at the same time, a significant technical undertaking. Recent progress in robotic hardware, artificial intelligence, and accelerated computing has collectively paved the ground for developing general-purpose robot autonomy. To march toward human-level physical intelligence, we advocate for a full-stack solution that integrates the three key ingredients: hardware, models, and data. First and foremost, robots are embodied physical agents, and their hardware determines their capability envelope. It makes **humanoid robots** a compelling form factor to build robot intelligence due to their human-like physique and versatility. Second, the diversity and variability of the real world demands that the robots operate on open-ended objectives and perform a wide range of tasks. Achieving this requires a **generalist robot model** sufficiently expressive and capable of handling various tasks. Third, real-world humanoid data are costly and time-consuming to acquire at scale. We need an effective data strategy to train large-scale robotic models.

In recent years, foundation models have brought forth dramatic breakthroughs in understanding and generating visual and text data. They demonstrate the effectiveness of training generalist models on web-scale data to enable strong generalization and fast adaptation to downstream tasks. The successes of foundation models in neighboring fields of AI have depicted a promising roadmap for building the "backbone" of intelligence for generalist robots, endowing them with a set of core competencies and enabling them to rapidly learn and adapt in the real world. However, unlike the digital realms of words and pixels, no Internet of humanoid robot datasets exist for large-scale pre-training. The data available for any single humanoid hardware would be orders of magnitude too small. Recent efforts in the robot learning community (Open

Figure 7.1: Data pyramid for robot foundation model training. GR00T N1's heterogeneous training corpora can be represented as a pyramid: data quantity decreases, and embodiment-specificity increases, moving from the bottom to the top.

X-Embodiment Collaboration et al., 2024) have explored cross-embodied learning to enlarge the dataset by pooling training data from many different robots. However, the great variability in robot embodiments, sensors, actuator degrees of freedom, control modes, and other factors result in an archipelago of "data islands" rather than a coherent, Internet-scale dataset needed for training a true generalist model.

We introduce GR00T N1, an open foundation model for generalist humanoid robots. The GR00T N1 model is a Vision-Language-Action (VLA) model, which generates actions from image and language instruction input. It has cross-embodiment support from tabletop robot arms to dexterous humanoid robots. It adopts a **dual-system compositional architecture**, inspired by human cognitive processing (Kahneman, 2011). The System 2 reasoning module is a pre-trained Vision-Language Model (VLM) that runs at 10Hz on an NVIDIA L40 GPU. It processes the robot's visual perception and language instruction to interpret the environment and understand the task goal. Subsequently, a Diffusion Transformer, trained with action flow-matching, serves as the System 1 action module. It cross-attends to the VLM output tokens and employs embodiment-specific encoders and decoders to handle variable state and action dimensions for motion generation. It generates closed-loop motor actions at a higher frequency (120Hz). Both the System 1 and System 2 modules are implemented as Transformer-based neural networks, tightly coupled and jointly optimized during training to facilitate coordination between reasoning and actuation.

To mitigate the "data island" problem mentioned earlier, we structure the VLA training corpora as a **data pyramid**, illustrated in Fig. 7.1. Rather than treating the training datasets as a homogeneous pool, we organize heterogeneous sources by scale: large quantities of web data and human videos lay the base of the pyramid; synthetic data generated with physics simulations and/or augmented by off-the-shelf neural models form the middle layer, and real-world data collected on the physical robot hardware complete the top. The lower layers of the pyramid provide broad visual and behavioral priors, while the upper layers ensure grounding in embodied, real-robot execution.

We develop an effective **co-training** strategy to learn across the entire data pyramid in both pre- and post-training phases. To train our model with action-less data sources, such as human videos and neural-generated videos, we learn a latent-action codebook (Ye et al., 2025) and also use a trained inverse dynamics model (IDM) to infer pseudo-actions. These techniques enable us to annotate actions on action-less videos so we can effectively treat them as additional robot embodiments for model training. By unifying all data sources across the data pyramid, we construct a consistent dataset where the input consists of the robot state, visual observations, and language instruction, and the output is the corresponding motor action. We pre-train our model end-to-end across the three data layers, spanning (annotated) video datasets, synthetically generated datasets, and real-robot trajectories — by sampling training batches across this heterogeneous data mixture.

With a unified model and single set of weights, GR00T N1 can generate diverse manipulation behaviors using single-arm, bimanual, and humanoid embodiments. Evaluated on standard simulation benchmark environments, GR00T N1 achieves superior results compared to state-of-the-art imitation learning baselines. We also demonstrate GR00T N1's strong performance in real-world experiments with GR-1 humanoid robots. Our GR00T-N1-2B model checkpoint, training data, and simulation benchmarks are publicly available here: GitHub and HuggingFace Datasets.

7.2 Related Work

Foundation Models in Robotics. Developing and using foundation models (Bommasani et al., 2021) for robotics has been of great interest recently. One common approach is to leverage existing pre-trained foundation models as high-level black-box reasoning modules in conjunction with low-level robot-specific policies (Brohan et al., 2023a; Huang et al., 2022b; Singh et al., 2023; Driess et al., 2023; Liang

et al., 2023; Lin et al., 2023; Huang et al., 2023b). This approach allows the robot to plan sequences of low-level skills or motions using the pre-trained foundation model. However, it assumes the availability of these low-level policies and a sufficient interface to connect them to the black-box foundation models. An alternative approach is to finetune pre-trained foundation models on robotics data to build Vision-Language-Action (VLA) models (Brohan et al., 2022; Brohan et al., 2023b; Black et al., 2024; Kim et al., 2024; Zheng et al., 2025; Wen et al., 2025; Cheang et al., 2024; Li et al., 2023; Zhen et al., 2024; Huang et al., 2024; Ye et al., 2025; Yang et al., 2025a). Instead of enforcing a rigid hierarchy between high-level VLM planning and low-level control, these VLA models allow for end-to-end optimization toward the downstream deployment tasks. We take a similar approach to train GR00T N1 and use the Eagle-2 model (Li et al., 2025) as our base Vision Language Model (VLM). We fine-tune our VLM together with a flow-matching (Lipman et al., 2022; Liu et al., 2022a; Hu et al., 2024) action generation model with action chunking (Zhao et al., 2023). In contrast to prior VLA models (Black et al., 2024) that use a mixture-of-experts architecture to bridge the base VLM model with the action generation model, we use a simple cross-attention mechanism. This approach provides flexibility regarding the exact architecture of the VLM model and the action generation model we can use. Furthermore, we use embodiment-specific state and action projector modules, which support different robot embodiments, including latent (Ye et al., 2025) and IDM-based (Baker et al., 2022) actions. The use of these projectors is similar to those in Octo Model Team et al. (2024), though that work did not fine-tune the VLM models.

Datasets for Robot Learning. A core challenge in robot learning is the scarcity of large-scale, diverse, and embodied datasets necessary to train generalist robots. One common approach is to use robot teleoperation (Zhang et al., 2018; Mandlekar et al., 2018; Mandlekar et al., 2019; Mandlekar et al., 2020; Wu et al., 2023b; Zhao et al., 2023; Aldaco et al., 2024; Fu et al., 2024; Iyer et al., 2024; Dass et al., 2024), where a human uses a device such as a smartphone or Virtual Reality (VR) controller, to control a robot to perform tasks of interest. The robot sensor streams and robot controls during operation are logged to a dataset, allowing for high-quality task demonstrations to be collected. Recently, this approach has been scaled by utilizing large teams of human operators and robot fleets over extended periods of time (e.g., months), resulting in large-scale robot manipulation datasets with thousands of hours of demonstrations (Ebert et al., 2022; Brohan et al., 2022; Ahn et al., 2022; Lynch et al., 2023; O'Neill et al., 2024; AgiBot-World-Contributors et al., 2025; Black et al.,

2024). However, collecting data this way requires extensive cost and human effort. Another line of work, instrumented human demonstrations, uses special hardware to capture robot-relevant observation and action data without explicitly teleoperating the target robot. For example, Chi et al., 2024b use hand-held robot grippers, Fang et al. (2024) uses a robot-like exoskeleton, and Kareer et al., 2024 uses special glasses to capture human hand motions, which are retargeted to robot action data. These approaches tend to result in faster data collection, though they have a mismatch with the downstream robot compared to direct robot teleoperation. A separate line of work makes use of human video datasets (Grauman et al., 2024; Grauman et al., 2022; Goyal et al., 2017; Damen et al., 2018; Miech et al., 2019), which are plentiful and substantially easier to collect than on-robot data, as a source of training data for robots. Some works (Nair et al., 2022; Wu et al., 2023a; Karamcheti et al., 2023) use human video datasets to pre-train representations that are then used as a feature space for training policies on downstream robot datasets. Other works Bharadhwaj et al., 2024a; Bharadhwaj et al., 2024b; Ren et al., 2025a try to jointly use human video data and robot data through intermediate representations for the motions in the video. Ye et al. (2025) shows that pretraining VLAs with *latent* actions only on human videos yields positive transfer to downstream robotic tasks. Rather than relying on a single type of training data, we developed techniques to effectively learn from a diverse assortment of real-world robot data, human video data, and synthetic data.

Synthetic Data Generation in Robotics. Real-world robot data collection requires large amounts of time and considerable human cost. By contrast, data collection in simulation can be substantially more efficient and less painful, making it a compelling alternative. Recently, several works (Mandlekar et al., 2023; James et al., 2020; Dalal et al., 2023; Gu et al., 2023; Ha et al., 2023; Nasiriany et al., 2024; Jiang et al., 2024; Wang et al., 2024a; Garrett et al., 2024; Yang et al., 2025b) have proposed automated data generation pipelines that can leverage simulation to produce thousands of task demonstrations with minimal human effort. This makes it easy to generate large-scale datasets; however, utilizing these datasets can be challenging due to the simulation-to-reality gap.

Another promising avenue has been using neural generative models to augment existing sets of robot demonstrations (Mandi et al., 2022; Yu et al., 2023a; Chen et al., 2023). However, previous work have been limited to utilizing in-painting or text-to-image diffusion models to augment the training data. In our work, we

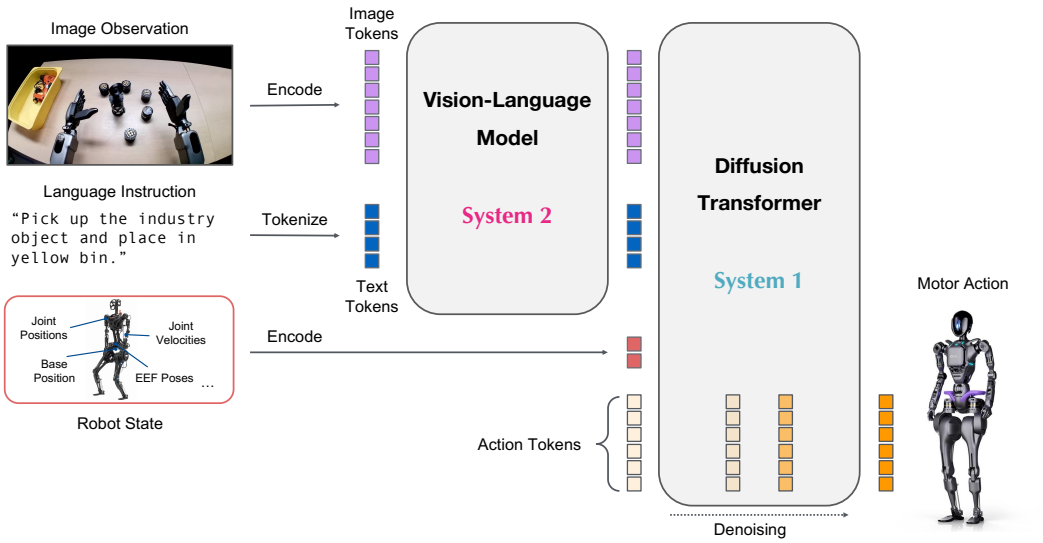


Figure 7.2: GR00T N1 model overview. Our model is a Vision-Language-Action (VLA) model that adopts a dual-system design. We convert the image observation and language instruction into a sequence of tokens to be processed by the Vision-Language Model (VLM) backbone. The VLM outputs, together with robot state and action encodings, are passed to the Diffusion Transformer module to generate motor actions.

leverage the recent advancements in video generative models (Agarwal et al., 2025; Wan Team, 2025) to create entire neural trajectories, at a scale that has never been explored before: ∼300k neural trajectories which amounts to 827 hours of robot trajectories.

In our model, we make use of large synthetic simulation datasets generated by MimicGen (Mandlekar et al., 2023) and DexMimicGen (Jiang et al., 2024), as well as neural-generated video datasets with state-of-the-art video generation models. Our way of co-training with synthetically generated and real-world data sets us from other large-scale VLA efforts.

7.3 GR00T N1 Foundation Model

GR00T N1 is a Vision-Language-Action (VLA) model for humanoid robots trained on diverse data sources. The model contains a vision-language backbone that encodes language and image input and a DiT-based flow-matching policy that outputs high-frequency actions. We use the NVIDIA Eagle-2 VLM (Li et al., 2025) as the vision-language backbone. Specifically, our publicly released GR00T-N1-2B model has 2.2B parameters in total, with 1.34B in the VLM. The inference time for sampling a chunk of 16 actions is 63.9ms on an L40 GPU using bf16. Fig. 7.2

provides a high-level overview of our model design. We highlight three key features of GR00T N1:

- We design a compositional model that integrates Vision-Language Model (VLM)-based reasoning module (System 2) and Diffusion Transformer (DiT)-based action module (System 1) in a unified learning framework;
- We develop an effective pre-training strategy using a mixture of human videos, simulation and neural-generated data, and real robot demonstrations (see Fig. 7.1) for generalization and robustness;
- We train a massively multi-task, language-conditioned policy that supports a wide range of robot embodiments and enables rapid adaptation to new tasks through data-efficient post-training.

Model Architecture

In this section, we describe the GR00T N1 model architecture, illustrated in Fig. 7.3. GR00T N1 uses flow-matching (Lipman et al., 2022) to learn action generation. A diffusion transformer (DiT) processes the robot's proprioceptive state and action, which are then cross-attended with image and text tokens from the Eagle-2 VLM backbone to output the denoised motor actions. Below, we elaborate on each module in detail.

State and Action Encoders To process states and actions of varying dimensions across different robot embodiments, we use an MLP per embodiment to project them to a shared embedding dimension as input to the DiT. As in Black et al. (2024), the Action Encoder MLP also encodes the diffusion timestep together with the noised action vector.

We use action flow matching, which samples actions through iterative denoising. The model takes as input noised actions in addition to encodings of the robot's proprioceptive state, image tokens, and text tokens. The actions are processed in chunks as in Zhao et al. (2023), meaning that at any given time t the model uses $A_t = [a_t, a_{t+1}, \ldots, a_{t+H-1}]$ which contains the action vectors of timesteps t through $t + H - 1$. We set $H = 16$ in our implementation.

Vision-Language Module (System 2) For encoding vision and language inputs, GR00T N1 uses the Eagle-2 (Li et al., 2025) vision-language model (VLM) pretrained

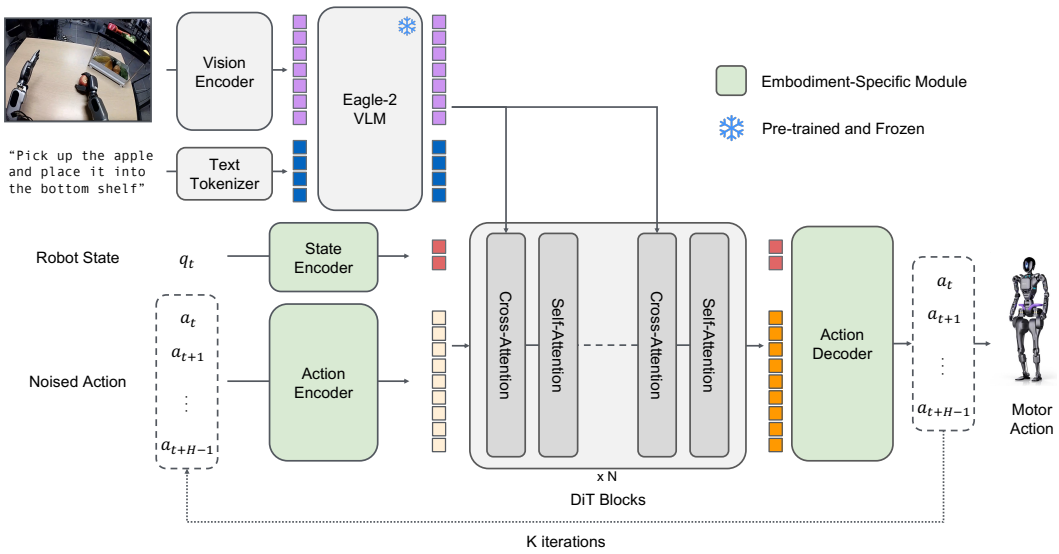


Figure 7.3: GR00T N1 model architecture. GR00T N1 is trained on a diverse set of embodiments ranging from single-arm robot arms to bimanual humanoid dexterous hands. To deal with different robot embodiment's state observation and action, we use DiT blocks with an embodiment-aware state and action encoder to embed the robot's state and action inputs. GR00T N1 model leverages latent embeddings of the Eagle-2 model to incorporate the robot's visual observation and language instructions. The vision language tokens will then be fed into the DiT blocks through cross-attention layers.

on Internet-scale data. Eagle-2 is finetuned from a SmolLM2 (Allal et al., 2025) LLM and a SigLIP-2 (Tschannen et al., 2025) image encoder. Images are encoded at resolution 224×224 followed by pixel shuffle (Shi et al., 2016), resulting in 64 image token embeddings per frame. These embeddings are then further encoded together with text by the LLM component of the Eagle-2 VLM. The LLM and image encoder are aligned over a broad set of vision-language tasks following the general recipe of Li et al. (2025).

During policy training, a text description of the task, as well as (possibly multiple) images, are passed to the VLM in the chat format used during vision-language training. We then extract vision-language features of shape (batch size $\times$ sequence length $\times$ hidden dimension) from the LLM. We found that using middle-layer instead of final-layer LLM embeddings resulted in both faster inference speed and higher downstream policy success rate. For GR00T-N1-2B, we use the representations from the 12th layer.

Diffusion Transformer Module (System 1) For modeling actions, GR00T N1 uses a variant of DiT (Peebles et al., 2023), which is a transformer with denoising step conditioning via adaptive layer normalization, denoted as V_θ. As shown in Fig. 7.3, V_θ consists of alternating cross-attention and self-attention blocks, similar to Flamingo (Alayrac et al., 2022) and VIMA (Jiang et al., 2023). The self-attention blocks operate on noised action token embeddings A_t^τ together with state embeddings q_t, while cross-attention blocks allow conditioning on the vision-language token embeddings ϕ_t output by VLM. After the final DiT block, we apply an embodiment-specific Action Decoder, another MLP, to the final H tokens to predict the actions.

Given a ground-truth action chunk A_t, a flow-matching timestep $\tau \in [0, 1]$ and sampled noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the noised action chunk A_t^τ is computed as $A_t^\tau = \tau A_t + (1 - \tau)\epsilon$. The model prediction $V_\theta(\phi_t, A_t^\tau, q_t)$ aims to approximate the denoising vector field $\epsilon - A_t$ by minimizing the following loss:

$$\mathcal{L}_{fm}(\theta) = \mathbb{E}_\tau \left[\| V_\theta(\phi_t, A_t^\tau, q_t) - (\epsilon - A_t) \|^2 \right]. \tag{7.1}$$

As in Black et al. (2024), we use $p(\tau) = \text{Beta}(\frac{s-\tau}{s}; 1.5, 1)$, $s = 0.999$. During inference, we generate action chunks with K-step denoising. First, randomly sample $A_t^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then use forward Euler integration to iteratively generate the action chunk, updating as follows:

$$A_t^{\tau+1/K} = A_t^\tau + \frac{1}{K} V_\theta(\phi_t, A_t^\tau, q_t).$$

In practice, we found $K = 4$ inference steps to work well across all embodiments.

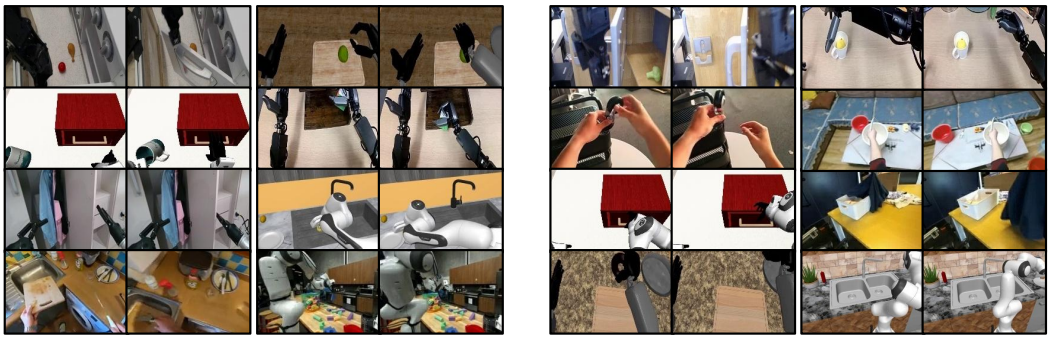
Training Data Generation

To train GR00T N1, we use a diverse set of data sources and objectives to construct the data pyramid (Fig. 7.1). We first source diverse human egocentric video data from open datasets, which forms the base, together with the web data used in VLM pretraining. Next, we generate synthetic *neural* trajectories using pre-trained video generation models. In this way, we $\sim$10$\times$ our in-house collected teleoperation trajectories — the "peak" of the data pyramid — from 88 hours to 827 hours, using diverse counterfactual robot trajectories with novel language instructions (see Fig. 7.5 for examples). We additionally generate diverse simulation trajectories, which also expand the middle part of the data pyramid.

In the next paragraph, we first describe how we extract *latent* actions from videos, which we use to extract labels for web-scaled human egocentric datasets. Next, we

describe how we generate *neural* and *simulated* robot trajectories, and how we obtain actions for each of these data sources.

Latent Actions For human egocentric videos and neural trajectories, we do not have any actions that we can directly use to train GR00T N1. For these data, we instead generate latent actions by training a VQ-VAE model to extract features from consecutive image frames from videos (Ye et al., 2025). The encoder takes the current frame x_t and the future frame x_{t+H} of a video with a fixed window size H and outputs the latent action z_t. The decoder is trained to take the latent action z_t and x_t and reconstruct x_{t+H}. This model is trained with a VQ-VAE objective, where the continuous embedding from the encoder is mapped to the nearest embedding from the codebook. After training, we take the encoder and use it as an inverse dynamics model; given an x_t and x_{t+H} pair, we extract the continuous pre-quantized embedding and use this as the latent action label during pre-training, with the same flow-matching loss, but treat it as a distinct "LAPA" embodiment. Training the VQ-VAE model on all heterogeneous data together allows us to unify all of the data to share the same learned latent action space, potentially improving cross-embodiment generalization. Fig. 7.4 shows x_t and x_{t+H} pairs from 8 distinct embodiments including both robot and human embodiment, all retrieved from similar latent actions; the first latent action shows all embodiments *moving right arm to the left* and the second latent action shows *moving right arm to the right*.



Latent action 1: Move the right arm to the left

Latent action 2: Move the right arm to the right

Figure 7.4: Latent actions. We retrieve similar latent embeddings across various embodiments. The left images illustrate the latent action that corresponds to moving the right arm (or hand) to the left, while the right images illustrate the latent action that corresponds to moving the right arm (or hand) to the right. Note that this general latent action is not only consistent in different robot embodiments, but also in human embodiment.

Neural Trajectories Robot data scales linearly with human labor, since it typically requires a human operator to teleoperate the robot to produce each trajectory. Recently, **video generation models** have demonstrated significant potential for high-quality controllable video generation (Brooks et al., 2024; Yang et al., 2024; Xiang et al., 2024; Lin et al., 2024; Wan Team, 2025; Ren et al., 2025b), which paves the way for building world models in the robotic domain. To harness these models, we fine-tune image-to-video generation models (Agarwal et al., 2025; Yang et al., 2024; Wan Team, 2025) on all of our 88 hours of in-house collected teleoperation data and generate 827 hours of video data given the existing initial frames with novel language prompts, augmenting it by around $10\times$. This enables generating training data that captures many more counterfactual scenarios in the real world without actually collecting teleoperation data for each of these cases (examples shown in Fig. 7.5.

To increase the diversity of our neural trajectories, we first use a commercial-grade multimodal LLM to detect the objects given initial frames and generate many more possible combinations of "*pick up {object} from {location A} to {location B}*," while instructing the model to only consider the physically feasible combinations. We also apply post-processing mechanisms, including filtering and re-captioning, to the generated videos. For this, we also use a commercial-grade multimodal LLM as a judge and feed the downsampled 8 frames to filter out neural trajectories that do not follow the language instruction precisely. We then caption the filtered-out videos.

Simulation Trajectories Scaling up real-world data collection for humanoid robots is highly expensive due to the challenge of simultaneously controlling both arms and dexterous hands. Recent research (Wang et al., 2024a; Mandlekar et al., 2023; Jiang et al., 2024) has demonstrated that generating training data in simulation is a practical alternative. We use DexMimicGen (Jiang et al., 2024) to synthesize large-scale robot manipulation trajectories.

Starting with a small set of human demonstrations, DexMimicGen applies demonstration transformation and replay in simulation to expand the dataset automatically. Each task is decomposed into a sequence of object-centric subtasks. The initial human demonstrations are segmented into smaller manipulation sequences, each corresponding to a subtask involving a single object. These segments are then adapted to new environments by aligning them with the object's position, preserving the relative poses between the robot's end effector and the object. To ensure smooth

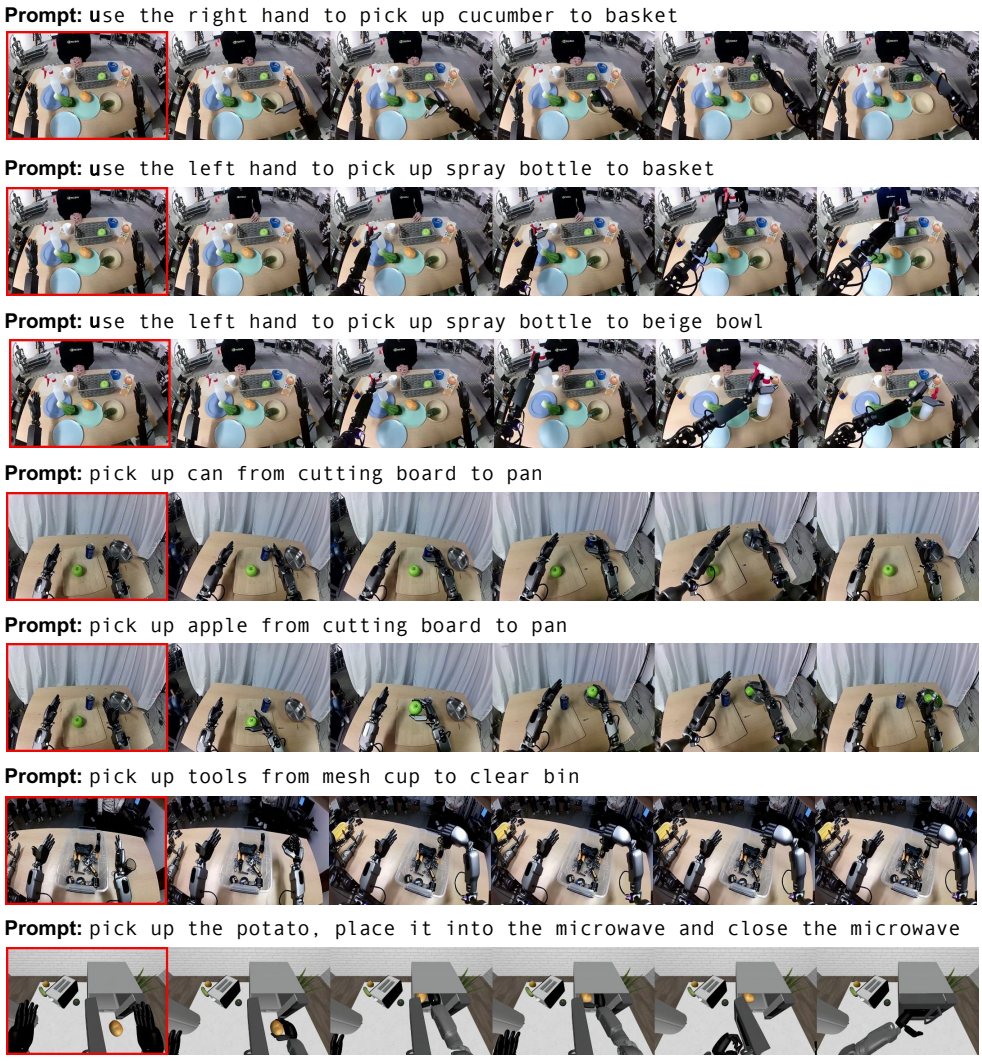


Figure 7.5: Synthetically generated videos. We leverage off-the-shelf video generation models to create neural trajectories to increase the quantity and diversity of our training datasets. These generated data can be used for both pre- and post-training of our GR00T N1. (1) The first three rows are generated from the same initial frames but with different prompts (change left or right, the location to place the object), (2) the following two are from the same initial frames but replace the object to pick up, (3) the next row showcases the video model generating a robot trajectory which is very challenging to generate in simulation (spilling contents inside a mesh cup into a bin), and (4) the last row is generated from an initial frame from simulation data. We use the red rectangles to indicate the initial frames.

execution, the system interpolates movements between the robot's current state and the transformed segment. The robot then follows the full sequence step by step, verifying task success at the end. Only successful demonstrations are retained, ensuring high-quality data. Using DexMimicGen, we scale a limited set of human demonstrations into a large-scale humanoid manipulation dataset. Considering the pre- and post-training datasets, we have generated 780,000 simulation trajectories — equivalent to 6,500 hours, or nine continuous months, of human demonstration data — in just 11 hours. These simulation data significantly supplement the real-robot data with minimal human costs.

Training Details

Pre-training During the pre-training phase, GR00T N1 is trained via flow-matching loss (Equation 7.1) on a diverse collection of embodiments and data sources, encompassing various real and synthetic robot datasets as well as human motion data. We refer readers to Sec. 7.4 for a detailed description of the datasets.

For human videos, in the absence of ground-truth actions, we extract learned latent actions and use them as flow-matching targets (see Sec. 7.3). For robot datasets such as our GR-1 humanoid data or Open X-Embodiment data, we use both ground-truth robot actions as well as learned latent actions as flow-matching targets. In the case of neural trajectories (Sec. 7.3) used to augment our robot datasets, we use both latent actions as well as predicted actions from an inverse-dynamics model trained on the real robot data.

Post-training In the post-training phase, we fine-tune our pre-trained model on datasets corresponding to each single embodiment. As in pretraining, we keep the language component of the VL backbone frozen and fine-tune the rest of the model.

Post-training with Neural Trajectories To overcome the challenge of data scarcity during post-training, we explore augmenting the data for each downstream task by generating neural trajectories, similar to the procedure described in Sec. 7.3. For downstream tasks that are conditioned on multiple views, we finetune the video model to generate multiple subimages in a grid. For simulation tasks, we collect diverse initial frames from the randomly initialized environment. For real robot tasks, we randomly initialize object poses manually and record the robot's initial observation. Novel initial frames could also be created automatically using img2img diffusion, but we leave further exploration for future work. We also demonstrate examples of

(1) multi-round video generation for generating long-horizon trajectories composed of atomic tasks and (2) neural trajectories of liquids and articulated objects, known to be extremely challenging to simulate, though we leave quantitative evaluation of downstream tasks for future work.

For our post-training pipeline with neural trajectories, we restrict ourselves to fine-tuning the video generation model *only* on the human-collected trajectories for simulation tasks and only 10% of the data from the real-world benchmark collected for post-training, to match the realistic scenario that we only have access to limited number of teleoperation data. Since the generated videos do not have action labels, we use either latent or inverse dynamics models (IDM) labeled actions (Baker et al., 2022) and train the policy model to treat these pseudo-actions as action labels for a different embodiment. In low-data regime scenarios, we also restrict ourselves on training the IDM models only on the low-data, to facilitate realistic scenarios. Some empirical comparisons between latent and IDM-labeled actions are made in Sec. 7.5. During post-training, we co-train the policy with real-world trajectories with neural trajectories with a 1:1 sampling ratio.

Training Infrastructure We train GR00T N1 on a cluster managed via NVIDIA OSMO (NVIDIA, 2025), an orchestration platform for scaling complex robotics workloads. The training cluster is equipped with H100 NVIDIA GPUs connected via NVIDIA Quantum-2 InfiniBand in a fat-tree topology. We facilitate fault-tolerant multi-node training and data ingestion via a custom library built on top of the Ray distributed computing library (Moritz et al., 2018). We use up to 1024 GPUs for a single model. GR00T-N1-2B used roughly 50,000 H100 GPU hours for pretraining.

Compute-constrained finetuning was tested in the context of a single A6000 GPU. If only tuning the adapter layers (action and state encoders + action decoder) and DiT, a batch size up to 200 can be used. When tuning the vision encoder, a batch size of up to 16 can be used.

7.4 Pre-Training Datasets

We structure our pre-training corpus into three main categories: real-robot datasets (Sec. 7.4), synthetic datasets (Sec. 7.4), and human video datasets (Sec. 7.4). These roughly correspond to the peak, middle, and base of the data pyramid (Fig. 7.1), respectively. The synthetic datasets consist of both simulation trajectories and neural trajectories. Table 7.1 summarizes our training data generation strategies in Sec. 7.3 and their applicable data sources correspondingly.

Table 7.1: Training data generation. Our data generation strategies leverage different data sources. The latent-action learning technique is broadly applied to diverse video datasets. Neural trajectories can be generated from datasets containing robot actions, while simulation trajectories rely on a physics simulator and utilize our DexMimicGen-based automated data generation system.

	Latent Actions	Neural Trajectories	Simulation Trajectories
Real-Robot Datasets	✓	✓	✓
Simulated Robot Datasets	✓	✓	
Human Video Datasets	✓		

Real-World Datasets

We use the following real-world robot datasets:

1. **GR00T N1 Humanoid Pre-Training Dataset.** Our internally collected dataset covers a broad range of general manipulation tasks, focused on Fourier GR1 through teleoperation. We leverage the VIVE Ultimate Tracker to capture the teleoperator's wrist poses while Xsens Metagloves track finger movements. We also explored other teleoperation hardware options, including Apple Vision Pro and Leap Motion (see Fig. 7.6). The recorded human movements are then retargeted to humanoid actions via inverse kinematics. The real-time teleoperation operates at a control frequency of 20Hz. Alongside the robot's actions, we capture images from a head-mounted camera at each step, as well as the human's low-dimensional proprioception and actions. The dataset includes fine-grained annotations, which detail atomic actions such as grasping, moving, and placing, and coarse-grained annotations, which aggregate sequences of fine-grained actions into higher-level task representations. This hierarchical structure supports learning both precise motion control and high-level task reasoning.

2. **Open X-Embodiment.** Open X-Embodiment Collaboration et al. (2024) is a widely used cross-embodiment dataset for robot manipulation. We include the RT-1 (Brohan et al., 2022), Bridge-v2 (Walke et al., 2023), Language Table (Lynch et al., 2023), DROID (Khazatsky et al., 2024), MUTEX (Shah et al., 2023), RoboSet (Bharadhwaj et al., 2024c) and Plex (Thomas et al., 2023), providing diverse datasets covering various manipulation tasks, language-conditioned control, and robot-environment interactions.

3. **AgiBot-Alpha.** AgiBot-World-Contributors et al. (2025) is a large-scale dataset

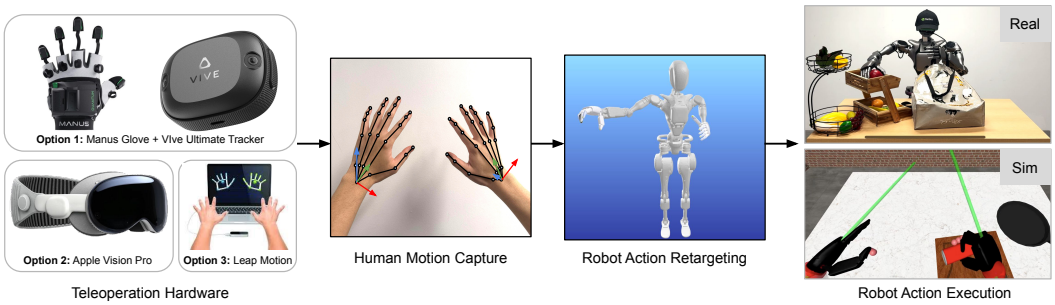


Figure 7.6: Data collection via teleoperation. Our teleoperation infrastructure supports multiple devices to capture human hand motion, including 6-DoF wrist poses and hand skeletons. Robot actions are produced through retargeting and executed on robots in real and simulation environments.

of trajectories from 100 robots. We used the 140,000 trajectories available at the time of launching our training run. The dataset covers fine-grained manipulation, tool usage, and multi-robot collaboration.

Synthetic Datasets

Our synthetic datasets include 1) simulation trajectories automatically multiplied from a small number of human demonstrations within physics simulators and 2) neural trajectories derived from videos produced by off-the-shelf neural generation models.

Simulation Trajectories In addition to real-world datasets, we feature large-scale synthetic datasets generated in simulation as described in Sec. 7.3. Our simulation tasks comprise humanoid robots performing a broad range of tabletop rearrangement tasks and feature a large array of realistic 3D assets. We build these tasks under the RoboCasa simulation framework (Nasiriany et al., 2024). Broadly, our tasks follow the behavior "rearrange A from B to C," where A corresponds to an object, and B and C represent the source and target locations in the environment. The source and target locations are receptacles such as plates, baskets, placemats, and shelves, and the robot must rearrange objects across different combinations of source and target receptacles. Overall, our pre-training simulation datasets feature 54 unique combinations of source and target receptacle categories. We place the objects and receptacles in randomized locations throughout the table and additionally incorporate distractor objects and receptacles in the scene. The distractors require the model to pay attention to the task language to perform the desired behavior.

We generate diverse, high-quality training datasets at a massive scale using DexMimicGen. Our datasets feature the GR-1 humanoid robot, but we can adopt the system for a wide range of robots. We begin by collecting a few dozen source demonstrations via teleoperation using the Leap Motion device. The Leap Motion device tracks the 6-DoF wrist poses and finger poses, and we retarget these values and send them to the whole-body IK controller based on mink (Zakka, 2024). Given human demonstrations, DexMimicGen processes the demonstrations into object-centric segments and then transforms and combines these segments to generate new demonstrations. Using this system, we generate 10,000 new demonstrations for each (source, target) receptacle pair in our pre-training task regime, resulting in 540k total demonstrations.

Neural Trajectories To generate neural trajectories, we fine-tune open-source image-to-video models on our real-world GR00T N1 Humanoid Pre-Training dataset, as described in Sec. 7.3. We trained the models for 100 epochs on a dataset comprising 3,000 real-world robot data samples with language annotations, each recorded at 480P resolution and consisting of 81 frames. As illustrated in Fig. 7.5, our model can generate high-quality counterfactual trajectories given novel language prompts. Moreover, the model, trained on Internet-scale video data, demonstrates strong generalization capabilities in handling unseen initial frames, novel objects, and new motion patterns. These videos are further labeled with latent actions and IDM-based pseudo-actions for model training. We generate a total of around 827 hours of videos; it takes 2 minutes to generate a one-second video on an L40 GPU, and required approximately 105k L40 GPU hours ($\sim$1.5 days) on 3,600 L40 GPUs.

Human Video Datasets

We include a diverse set of human video datasets. These do not include explicit action labels but contain extensive sequences of human-object interactions, capturing affordances, task semantics, and natural motion patterns. These datasets cover a wide range of real-world human behaviors, including grasping, tool use, cooking, assembly, and other task-oriented activities performed in natural environments, and provide detailed first-person perspectives of hand-object interactions. Our video datasets include the following:

- **Ego4D** is a large-scale egocentric video dataset that includes diverse recordings of everyday activities (Grauman et al., 2022);

- **Ego-Exo4D** adds complementary exocentric (third-person) views alongside first-person recordings (Grauman et al., 2024);
- **Assembly-101** focuses on complex assembly tasks by providing detailed videos of step-by-step object assembly (Sener et al., 2022);
- **EPIC-KITCHENS** includes first-person footage of culinary activities (Damen et al., 2018);
- **HOI4D** captures human-object interactions with frame-wise annotations for segmentation, hand and object poses, and actions (Liu et al., 2022b);
- **HoloAssist** captures collaborative and assistive tasks within augmented reality environments (Wang et al., 2023d);
- **RH20T-Human** includes recordings of fine-grained manipulation tasks with an emphasis on natural hand-object interactions across diverse real-world scenarios (Fang et al., 2023).

7.5 Evaluation

We evaluate our GR00T N1 models in a diverse set of simulated and real-world benchmarks. Our simulation experiments are conducted on three distinct benchmarks designed to systematically assess the effectiveness of our model across various robot embodiments and manipulation tasks. In our real-world experiments, we investigate the model's capability on a suite of tabletop manipulation tasks with the GR-1 humanoid robot. These experiments aim to demonstrate GR00T N1's ability to acquire new skills from a limited number of human demonstrations.

Simulation Benchmarks

Our simulation experiments comprise two open-source benchmarks from prior work (Nasiriany et al., 2024; Jiang et al., 2024), as well as a newly developed suite of tabletop manipulation tasks designed to closely mirror our real-world task settings. We meticulously choose these benchmarks for evaluating our models across different robot embodiments and diverse manipulation tasks. Our model checkpoints, together with the publicly available simulation environments and datasets, ensure the reproducibility of our key results. Fig. 7.7 illustrates some example tasks from these three benchmarks.

- **RoboCasa Kitchen (24 tasks, RoboCasa)**

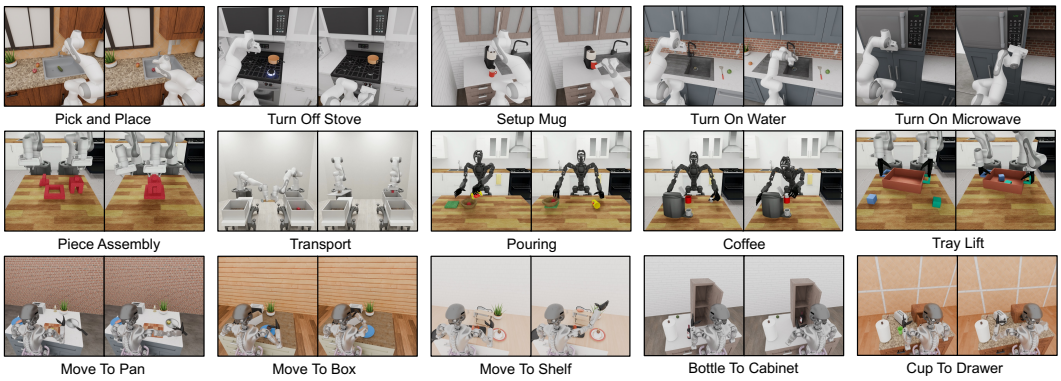


Figure 7.7: Simulation tasks. Our simulation experiments use tasks from two open-source benchmarks (RoboCasa (Nasiriany et al., 2024) in the top row and DexMimicGen (Jiang et al., 2024) in the middle row) and a newly developed suite of tabletop manipulation tasks that closely resemble our real-world tasks (bottom row). We provide Omniverse renderings of the tasks above.

RoboCasa (Nasiriany et al., 2024) features a collection of tasks in simulated kitchen environments. We focus on 24 "atomic" tasks that involve foundational sensorimotor skills such as pick-and-place, door opening and closing, pressing buttons, turning faucets, and more. For each task, we use the publicly available dataset of 3000 demonstrations featuring the Franka Emika Panda arm, all generated with MimicGen (Mandlekar et al., 2023). The observation space includes three RGB images captured from cameras positioned on the left, right, and at the wrist. The state representation comprises the position and rotation of both the end-effector and the robot base, as well as the gripper's state. The action space is defined by the relative position and rotation of the end-effector along with the gripper state. We follow the same training and evaluation protocol outlined by Nasiriany et al., 2024.

- **DexMimicGen Cross-Embodiment Suite (9 tasks, DexMG)**

  DexMimicGen (Jiang et al., 2024) includes an array of nine bimanual dexterous manipulation tasks requiring precise two-arm coordination. Together, these tasks cover three bi-manual robot embodiments: (1) *Bimanual Panda Arms with Parallel-Jaw Grippers*: tasks include threading, piece assembly, and transport. The state/action space consists of the end-effector position and rotation of both arms, as well as the gripper states; (2) *Bimanual Panda Arms with Dexterous Hands*: tasks include box cleanup, drawer cleanup, and tray lifting. The state/action space consists of the end-effector position and rotation of both arms and hands; (3) *GR-1 Humanoid with Dexterous Hands*: tasks

include pouring, coffee preparation, and can sorting. The state/action space consists of the joint position and rotation of both arms and hands, along with the waist and neck. We generate 1000 demonstrations for each task using the DexMimicGen data generation system and evaluate the model's ability to generalize to novel object configurations.

- **GR-1 Tabletop Tasks (24 tasks, GR-1)**

  This dataset serves as a digital counterpart to real-world humanoid datasets, enabling systematic evaluations that inform the performance of real-robot deployment. This benchmark focuses on dexterous hand control using the GR-1 humanoid robot equipped with Fourier dexterous hands. Compared to DexMG, this benchmark features a significantly larger variety of objects with diverse placements. We model a total of 18 rearrangement tasks, which have a similar structure to the pre-training tasks outlined in Sec. 7.4, i.e., rearranging objects from a source to a target receptacle. Each task involves a unique combination of receptacles, and these combinations are unseen in our pre-training data. Like the pre-training tasks, most tasks involve distractor objects and receptacles that require the model to pay attention to the task language. We additionally feature six tasks that involve placing objects into articulated objects (i.e., cabinets, drawers, and microwaves) and closing them. The observation space includes one RGB image captured from an egocentric camera positioned on the robot's head. The state/action space consists of the joint position and rotation of both arms and hands, along with the waist and neck. We optionally include in our datasets the end effector-based actions for controlling the arms, as the native action space for controlling the whole-body IK controller is end effector-based. We generate 1000 demonstrations for each task using the DexMimicGen system.

Real-World Benchmarks

We introduce a diverse and meticulously designed set of tabletop manipulation tasks, aimed at evaluating and post-training our models on human demonstrations. These tasks emphasize critical aspects of real-world dexterity, including precise object manipulation, spatial reasoning, bimanual coordination, and multi-agent collaboration. We carefully categorize our benchmarks into four distinct types, ensuring a rigorous evaluation of model performance. We show some example tasks from our real-world benchmarks in Fig. 7.8.

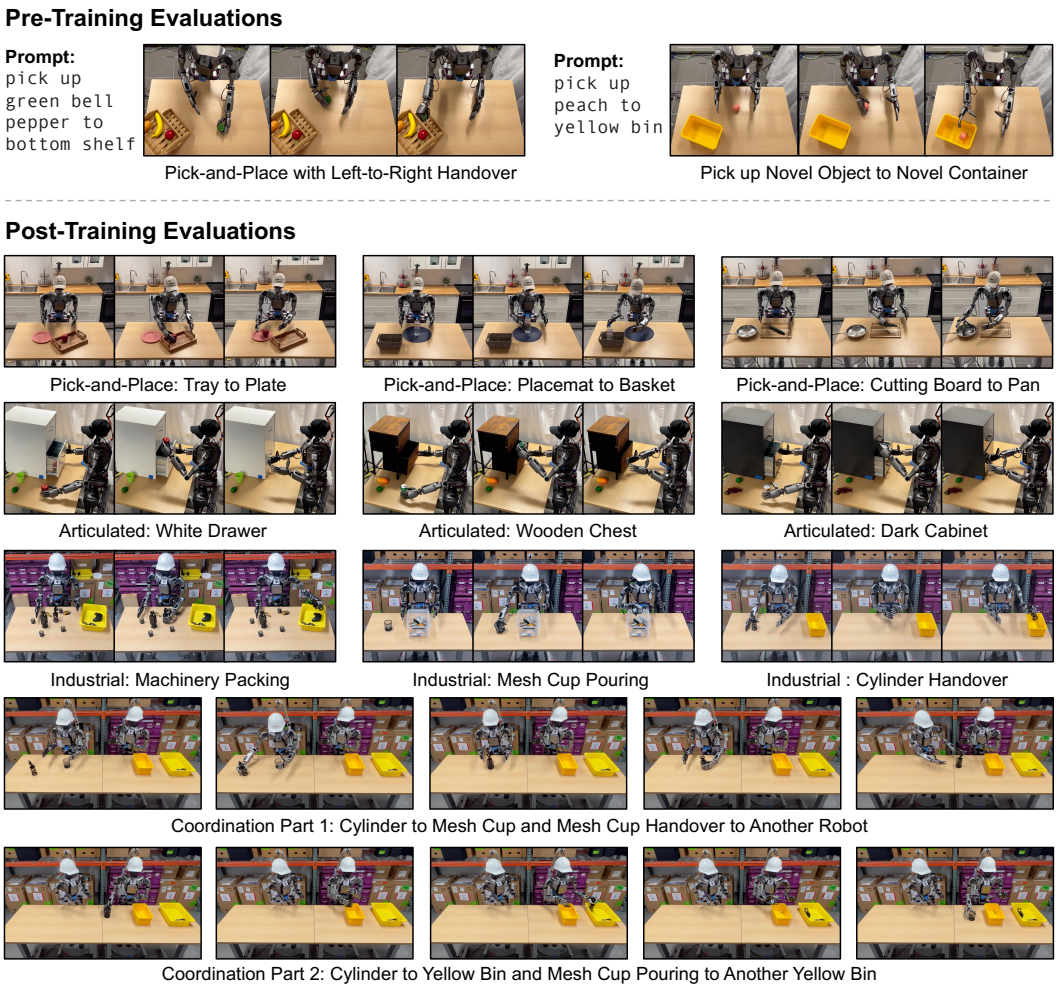


Figure 7.8: Real-world tasks. All images are captured from policy rollouts of GR00T-N1-2B and models post-trained from GR00T-N1-2B. (Top) Pre-training evaluations. We design two manipulation tasks to assess our pretrained models. The left image shows a left-to-right handover, while the right image illustrates the placement of novel objects into an unseen target container. (Bottom) Post-training evaluations. We introduce four distinct task categories. From top to bottom, we present examples of object-to-container pick-and-place, articulated object manipulation, industrial object manipulation, and multi-agent coordination.

- **Object-to-Container Pick-and-Place (5 tasks, Pick-and-Place)**

  This category evaluates the model's ability to grasp objects and place them into designated containers, a fundamental capability for robotic manipulation. Tasks include transferring objects between common household containers such as trays, plates, cutting boards, baskets, placemats, bowls, and pans. These scenarios test fine motor skills, spatial alignment, and adaptability to different object geometries. To rigorously assess generalization, we evaluate models on

both seen and unseen objects.

- **Articulated Object Manipulation (3 tasks, Articulated)**

  These tasks assess the model's ability to manipulate articulated storage compartments. The model must grasp an object, place it into a storage unit such as a wooden chest, dark cabinet, or white drawer, and then close the compartment. These tasks introduce challenges in constrained motion control and precise placement within limited spaces. Generalization is tested with both seen and unseen objects.

- **Industrial Object Manipulation (3 tasks, Industrial)**

  We design this category for industrial scenarios, which involve three structured workflows and tool-based interactions: 1) *Machinery Packing*: Pick up various machinery parts and tools and place them into a designated yellow bin; 2) *Mesh Cup Pouring*: Grasp a mesh cup containing small industrial components (e.g., screws and bolts) and pour its contents into a plastic bin; and 3) *Cylinder Handover*: Pick up a cylindrical object, transfer it from one hand to the other, and place it into a yellow bin. These tasks closely mirror real-world industrial applications, making them highly relevant benchmarks for assessing dexterity in structured environments.

- **Multi-Agent Coordination (2 tasks, Coordination)**

  Collaborative tasks require synchronization between multiple agents, emphasizing role coordination and adaptive decision-making: 1) *Coordination Part 1*: Pick up a cylinder, place it into a mesh cup, and hand it over to another robot; and 2) *Coordination Part 2*: The receiving robot places the cylinder into one yellow bin, then pours the remaining contents of the mesh cup into another yellow bin.

These carefully designed benchmarks introduce structured, goal-driven interactions to test whether a model can seamlessly adapt to real-world applications. To build a high-quality post-training dataset, we let human operators collect task-specific data for durations ranging from 15 minutes to 3 hours, depending on task complexity. We then filter out low-quality trajectories to maintain data integrity. By incorporating a diverse set of task requirements — spanning precise single-agent manipulation to complex multi-agent coordination—our benchmark provides a rigorous testbed

for evaluating generalization, adaptability, and fine-tuned control in human-like manipulation tasks.

Experiment Setup

Our evaluation experiment consists of post-training GR00T N1 and baseline models as described in Sec. 7.3 in a data-limited setting and evaluating the policy success rate in our simulated and real benchmarks described in Sections 7.5 and 7.5, respectively. By default we use a global batch size of 1024 and train for 60k steps. For the DexMimicGen Cross-Embodiment Suite, where each embodiment contains relatively few tasks and the overall training data is limited, we used a smaller batch size of 128 for GR00T-N1-2B.

Baselines To demonstrate the effectiveness of diverse pretraining of GR00T N1, we compare with two established baselines, BC-Transformer (Mandlekar et al., 2021) and Diffusion Policy (Chi et al., 2024a). We describe the details of these two methods below:

- **BC-Transformer** is a Transformer-based behavior cloning policy in RoboMimic (Mandlekar et al., 2021). It consists of a Transformer architecture for processing observation sequences and a Gaussian Mixture Model (GMM) module for modeling action distributions. The policy takes 10 observation frames as input and predicts the next 10 actions.
- **Diffusion Policy** (Chi et al., 2024a) models action distributions through a diffusion-based generative process. It employs a U-Net architecture that progressively removes noise from random samples to generate precise robot actions conditioned on observation sequences. It takes a single frame of observations as input and produces 16 action steps in one inference pass.

Evaluation Protocol For simulated benchmark evaluation, we report the average success rate over 100 trials, taking the maximum score of the last 5 checkpoints, where checkpoints are written every 500 training steps, following the protocol from RoboCasa (Nasiriany et al., 2024).

For real robot evaluation, we employ a partial scoring system to capture model behavior across different execution phases, ensuring a fine-grained assessment of performance. We report the average success rate over 10 trials for each task, except for the task of *Pack Machinery*; for this task, we report the success rate of how

many objects out of the 5 machinery parts and tools are placed into the bin, given a time-limit of 30 seconds. We conduct only 5 trials due to the time constraint. Additionally, to assess the model's efficiency in a low-data regime, we subsample 10% of the full dataset for each task and evaluate whether the model can still learn effective behaviors.

Quantitative Results

Pre-training Evaluations To evaluate the generalization capabilities of our pre-trained checkpoint, we design two tasks on the real GR-1 humanoid robot (Fig. 7.8). In the first task, the robot is instructed to place an object on the bottom shelf. However, the object is intentionally positioned to the left of its left hand, requiring a coordinated bimanual strategy. The robot must first grasp the object with its left hand, transfer it within reach of the right hand, and then complete the placement onto the shelf. In the second task, the robot is instructed to place a novel object into an unseen target container. For each task, we evaluate the pretrained GR00T-N1-2B model using five different objects, with three trials per object. GR00T-N1-2B achieves a success rate of 76.6% (11.5/15) in the first coordinated setting and 73.3% (11/15) in the second setting involving novel object manipulation. 0.5 stands for grasping the object correctly but failing to place the object into the container. The high performance under these two evaluation settings illustrates the effectiveness of large-scale pre-training.

Post-training Evaluations In simulation, we compare the quantitative results for our post-trained GR00T N1 models against from-scratch baselines in the three simulation benchmarks (Table 7.2). For each benchmark, we post-train using 30, 100, and 300 demonstrations per task (24 tasks for RoboCasa, 9 tasks for DexMG, and 24 tasks for GR-1). We observe that GR00T N1 consistently outperforms the baseline models across benchmark tasks and dataset sizes.

Table 7.2: Simulation results. Average success rate across three simulation benchmarks, using 100 demonstrations per task. GR00T N1 outperforms both baselines, especially on the GR-1 task where it outperforms by more than 17 %.

	RoboCasa	DexMG	GR-1	Average
BC Transformer	26.3%	53.9%	16.1%	26.4%
Diffusion Policy	25.6%	56.1%	32.7%	33.4%
GR00T-N1-2B	**32.1%**	**66.5%**	**50.0%**	**45.0%**

On the real robot, we compare GR00T-N1-2B against Diffusion Policy, training on 10% of the human teleoperation dataset and the full dataset (Table 7.3 and Fig. 7.9). GR00T-N1-2B, achieves a significantly higher success rate across all tasks, outperforming Diffusion Policy by 32.4% in the 10% Data setting and by 30.4% in the Full Data setting. Notably, GR00T-N1-2B trained on just 10% of the data performs only 3.8% lower than Diffusion Policy trained on the full dataset, highlighting its data efficiency.

Table 7.3: Real-world results. Average policy success rate on real-world tasks with the GR-1 humanoid robots. GR00T N1 beats the diffusion policy baseline and shows strong results even with very little data.

	Pick-and-Place	Articulated	Industrial	Coordination	Average
Diffusion Policy (10% Data)	3.0%	14.3%	6.7%	27.5%	10.2%
Diffusion Policy (Full Data)	36.0%	38.6%	61.0%	62.5%	46.4%
GR00T-N1-2B (10% Data)	35.0%	62.0%	31.0%	50.0%	42.6%
GR00T-N1-2B (Full Data)	**82.0**%	**70.9**%	**70.0**%	**82.5**%	**76.8**%

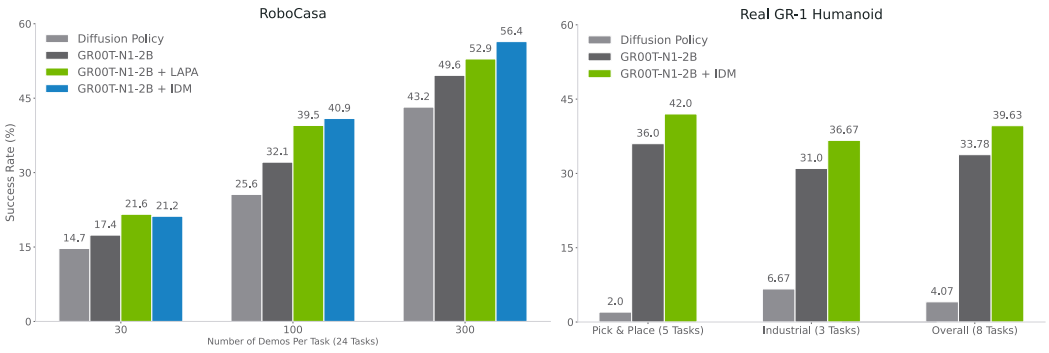


Figure 7.9: Neural trajectories ablations. In the RoboCasa simulation, we show using neural trajectories for post-training across 3 data regimes (30, 100, and 300 per task). In the real world, we show results only on the low-data regime (10% of the demonstrations). We co-train wit h 3k neural trajectories per task for RoboCasa and 100 neural trajectories per task for real-world tasks. We explore using both latent and IDM-labeled actions in simulation and only IDM-labeled actions for the real robot.

Post-training w/ Neural Trajectories Evaluations We show some preliminary results of using neural trajectories during post-training for the RoboCasa benchmark for simulation evaluation and Pick-and-Place (seen) and Industrial for the real-world evaluation in Figure 7.9. We observe that GR00T N1 co-trained with neural trajectories consistently results in substantial gains compared to GR00T N1 only trained on real-world trajectories: +4.2%, +8.8%, +6.8% on average for 30, 100, and

300 data-regimes, respectively, for RoboCasa and +5.8% on average across the 8 tasks with the GR-1 Humanoid.

When comparing LAPA and IDM labels in RoboCasa, an interesting pattern emerges: LAPA slightly outperforms IDM in the relatively low-data regime (30), but as more data becomes available (100 and 300), the performance gap between LAPA and IDM widens. This trend is intuitive — with more data for IDM training, the pseudo-action labels become increasingly aligned with real-world actions, leading to stronger positive transfer. Since GR-1 Humanoid is a relatively "high-data" regime for us, we only utilize IDM actions for neural trajectory co-training in the real world.

Qualitative Results

How does this behavior look qualitatively? To answer this, we consider the task "Turn Sink Spout" in RoboCasa — in the 100 sample regime, the DP baseline gets 11.8% success rate whereas GR00T N1 gets 42.2%. The DP baseline often gets confused about the semantics of the tasks. From Table 7.2, we see that GR00T N1 has strong results in the low-data regime. It is natural, in the limit of large fine-tuning datasets, that the effect of pre-training dwindles.

When prompting the pre-trained GR00T N1 model with the task instruction "Pick up the red apple and place it in the basket," one of the tasks in our post-training benchmark, we observe interesting behavioral patterns. In this scenario, we intentionally position the apple to the left of the humanoid hand. Despite seeing few similar tasks during pretraining and exhibiting jerkier motions, the pretrained checkpoint uses its left hand to grasp the apple, hands it over to the right hand, and then places it into the basket. In contrast, the post-trained checkpoint fails in this scenario. Since all post-training data exclusively involve the right hand without any inter-hand transfer, the post-trained policy loses the capability to perform this behavior.

For post-trained GR00T N1, we observed that, compared to the baseline Diffusion Policy, its motion is generally much smoother, and its grasping accuracy is significantly higher. In contrast, the Diffusion Policy baseline suffers from immobility during the initial frames and frequently exhibits inaccurate grasping, resulting in a low success rate in our real-world benchmarks.

Limitations

Currently, our GR00T N1 model focuses primarily on short-horizon tabletop manipulation tasks. In future work, we aim to extend its capabilities to tackle long-horizon

loco-manipulation, which will require advancements in humanoid hardware, model architecture, and training corpora. We anticipate a stronger vision-language backbone will enhance the model's spatial reasoning, language understanding, and adaptability. Our synthetic data generation techniques — leveraging video generation models and automated trajectory synthesis systems — have shown great promise. However, existing methods still face challenges in generating diverse and counterfactual data, while adhering to the laws of physics, limiting the quality and variability of synthetic datasets. We aim to enhance our synthetic data generation techniques to further enrich our data pyramid for model training. Furthermore, we plan to explore novel model architectures and pre-training strategies to improve the robustness and generalization capabilities of our generalist robot models.

7.6 Conclusions

We have presented GR00T N1, an open foundation model for generalist humanoid robots. GR00T N1 features a dual-system model design, leverages heterogeneous training data, and supports multiple robot embodiments. We systematically evaluate it as a generalist policy across simulation benchmarks and on the real GR-1 humanoid robot. Our experiments demonstrate its strong generalization capabilities, enabling robots to learn diverse manipulation skills with high data efficiency. We hope that our open GR00T-N1-2B model, alongside its training datasets and simulation environments, will accelerate the community's progress toward building and deploying generally capable humanoid robots in the wild.

Chapter 8

CONCLUSIONS AND FUTURE DIRECTIONS

8.1 Summary

This dissertation has examined a unified and scalable recipe for building foundation agents grounded in three core components: internet-scale multimodal data, LLMs as the embodied reasoning engine, and unified VLA architectures. Across six chapters, we have discussed how these ingredients jointly advance the development of embodied agents that learn continually, generalize broadly, and operate reliably in complex, open-world environments.

Chapters 2 and 3 focus on leveraging internet-scale multimodal data to endow agents with broad prior knowledge and diverse behavioral repertoires. Chapter 2 introduces MineDojo, a framework that combines an open-ended simulation environment with a large-scale database of videos, tutorials, and wiki knowledge to support general-purpose skill acquisition. Chapter 3 presents NitroGen, a video-action foundation model trained on 40,000 hours of gameplay videos across more than 1,000 games, demonstrating the effectiveness of large-scale behavior cloning for cross-game generalization and transfer.

Chapters 4 and 5 investigate LLMs as the reasoning engine within the perception–action loop. Chapter 4 describes Voyager, an LLM-powered lifelong learning agent that autonomously explores, builds a growing skill library, and continually improves through iterative prompting and environment feedback. Chapter 5 introduces Eureka, a framework that uses LLMs to generate and refine reward functions, enabling reinforcement learning agents to acquire complex manipulation skills and providing a gradient-free approach to reinforcement learning from human feedback.

Chapters 6 and 7 develop unified VLA architectures that couple perception, language grounding, and motor control. Chapter 6 proposes VIMA, a multimodal prompting formulation and transformer-based agent capable of handling a wide variety of tabletop manipulation tasks through a uniform sequence modeling interface. Chapter 7 presents GR00T N1, an open VLA foundation model for humanoid robots that integrates vision, language, and real-time action generation through a dual-system architecture, demonstrating strong performance across diverse embodiments and successful deployment on real humanoid platforms.

8.2 Future Directions

While this dissertation outlines a unified recipe for building foundation agents, several promising directions remain open for advancing general-purpose embodied intelligence even further. These directions highlight the need to move beyond imitation, leverage scalable sources of human behavior, incorporate autonomous exploration, and fully integrate language-driven reasoning into the lifelong learning process.

Real-World Data Flywheel for VLA-RL Post-Training

A major limitation of agents trained purely through internet-scale imitation learning is that they inherit the ceiling of the underlying behavior policy. To achieve competence beyond what is present in web videos or expert demonstrations, agents must be able to refine and improve their policies through autonomous interaction. This calls for a real-world *data flywheel* for RL atop a VLA foundation model (Amin et al., 2025). After an initial behavior cloning phase, the model should continue to learn from its own rollouts, discover new behaviors, and self-correct through iterative improvement. Achieving this will require a foundation reward model (Zhang et al., 2025) for robotics, analogous to MineCLIP for MineDojo, capable of assessing real-world trajectories, guiding exploration, and providing dense, scalable feedback signals without labor-intensive reward engineering. Such VLA-RL post-training represents a crucial step toward embodied agents that learn continually and surpass the limitations of internet-scale demonstrations.

RL in World Models for Efficient Autonomous Improvement

Despite its importance, real-world RL is costly, slow, and equipment-intensive. An appealing alternative is to perform most of the optimization inside a learned world model, where agents can explore and improve policies at simulation speed. Recent advances such as Dreamer 4 (Hafner et al., 2025) and Ctrl-World (Guo et al., 2025) illustrate the promise of RL in world models. Incorporating VLA architectures into model-based RL may yield powerful agents that combine the scalability of foundation models with the closed-loop efficiency of learned dynamics. A compelling direction is to develop world models tailored for robotics that capture not only physics, but also objects, affordances, and the semantic structure required for language-conditioned control. This would significantly reduce reliance on expensive real-world rollouts, enabling scalable self-improvement beyond imitation.

Learning from Web-Scale Human Activity Videos

Just as the internet-scale gaming videos enabled the emergence of generalist agents in MineDojo and NitroGen, web-scale human activity datasets represent a largely untapped resource for robot learning. Human videos contain rich demonstrations of manipulation, navigation, tool use, problem-solving strategies, and everyday task structure in natural environments. Future work may focus on extracting actionable information from these videos, including inferring fine-grained action labels, recovering 3D hand-object trajectories, or learning preference signals that reflect how humans judge success. Beyond direct imitation, these data may also serve as priors for reward modeling, affordance prediction, skill discovery, or high-level task scaffolding. Leveraging such vast, weakly labeled corpora would allow robotic agents to inherit much of humanity's procedural knowledge, greatly expanding their initial competence before interacting with the real world.

LLMs for Task Generation, Skill Discovery, and Autonomous Curricula

LLMs have already proven effective at generating rewards and improving agent behavior through self-refinement. A natural next step is to harness LLMs for autonomous task generation, environment design, and curriculum learning for robotics (Liang et al., 2024). Similar to the role LLMs played in Voyager, future systems may continuously propose new objectives, decompose skills, generate training scenarios, and identify gaps in the agent's capabilities. To realize this vision, robust and expressive simulation environments will be required (Mittal et al., 2025; Nasiriany et al., 2024), enabling LLMs to construct diverse tasks that are grounded in realistic physics and manipulable objects. Such closed-loop integration between LLM-driven reasoning, scalable simulation, and real-world execution could give rise to agents that continually expand their skill sets without human supervision.

8.3 Closing Thoughts

We are living in a remarkable moment for AI. The release of ChatGPT ignited global excitement by demonstrating how large-scale models, trained on broad data and simple objectives, can unlock emergent capabilities far beyond what was previously thought possible. In robotics, we are collectively working toward an analogous "ChatGPT moment," a breakthrough where embodied agents acquire general, reliable, and scalable competence in the physical world. Realizing this vision will require embracing the bitter lesson: scalable methods that leverage massive computation, large datasets, and end-to-end learning ultimately outperform hand-engineered

pipelines. The research in this dissertation contributes toward this direction by exploring how internet-scale multimodal data, language-based reasoning, and unified VLA architectures can serve as the foundation for general-purpose embodied agents. Although significant challenges remain, the trajectory of progress is accelerating, and the opportunities ahead are extraordinary. It is an exciting time to contribute to the future of robotics, and I am optimistic that scalable, principled approaches will bring us closer to embodied systems that assist and empower people in meaningful ways.

BIBLIOGRAPHY

Abbeel, Pieter and Andrew Y Ng (2004). Apprenticeship Learning via Inverse Reinforcement Learning. In: *Proceedings of the Twenty-First International Conference on Machine Learning*, p. 1.

Abramson, Josh et al. (2020). Imitating Interactive Intelligence. In: *arXiv preprint arXiv: Arxiv-2012.05672*.

Abu-El-Haija, Sami, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan (2016). YouTube-8M: A Large-Scale Video Classification Benchmark. In: *arXiv preprint arXiv: Arxiv-1609.08675*.

Aceituno, Bernardo, Alberto Rodriguez, Shubham Tulsiani, Abhinav Gupta, and Mustafa Mukadam (2021). A Differentiable Recipe for Learning Visual Non-Prehensile Planar Manipulation. In: *5th Annual Conference on Robot Learning*. URL: https://openreview.net/forum?id=f7KaqYLO3iE.

Agarwal, Niket, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. (2025). Cosmos World Foundation Model Platform for Physical AI. In: *arXiv preprint arXiv:2501.03575*.

Agarwal, Rishabh, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare (2021). Deep Reinforcement Learning at the Edge of the Statistical Precipice. In: *Advances in Neural Information Processing Systems* 34, pp. 29304–29320.

AgiBot-World-Contributors et al. (2025). AgiBot World Colosseo: A Large-Scale Manipulation Platform for Scalable and Intelligent Embodied Systems. In: *arXiv preprint arXiv:2503.06669*.

Agrawal, Pulkit (Nov. 2022). The Task Specification Problem. In: *Proceedings of the 5th Conference on Robot Learning*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, pp. 1745–1751. URL: https://proceedings.mlr.press/v164/agrawal22a.html.

Ahmed, Ossama, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Manuel Wuthrich, Yoshua Bengio, Bernhard Schölkopf, and Stefan Bauer (2021). CausalWorld: A Robotic Manipulation Benchmark for Causal Structure and Transfer Learning. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=SK7A5pdrgov.

Ahn, Michael, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. (2022). Do as I Can, Not as I Say: Grounding Language in Robotic Affordances. In: *arXiv preprint arXiv:2204.01691*.

Akbari, Hassan, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong (2021). VATT: Transformers for Multimodal Self-Supervised Learning From Raw Video, Audio and Text. In: *arXiv preprint arXiv: Arxiv-2104.11178*.

Alayrac, Jean-Baptiste, Adrià Recasens, Rosalia Schneider, Relja Arandjelovic, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman (2020). Self-Supervised MultiModal Versatile Networks. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. URL: https://proceedings.neurips.cc/paper/2020/hash/0060ef47b12160b9198302ebdb144dcf-Abstract.html.

Alayrac, Jean-Baptiste et al. (2022). Flamingo: A Visual Language Model for Few-Shot Learning. In: *arXiv preprint arXiv: Arxiv-2204.14198*.

Aldaco, Jorge, Travis Armstrong, Robert Baruch, Jeff Bingham, Sanky Chan, Kenneth Draper, Debidatta Dwibedi, Chelsea Finn, Pete Florence, Spencer Goodrich, et al. (2024). Aloha 2: An Enhanced Low-Cost Hardware for Bimanual Teleoperation. In: *arXiv preprint arXiv:2405.02292*.

Allal, Loubna Ben, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. (2025). SmolLM2: When Smol Goes Big–Data-Centric Training of a Small Language Model. In: *arXiv preprint arXiv:2502.02737*.

Amin, Ali, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, et al. (2025). $\Pi^*_{0.6}$: A VLA That Learns From Experience. In: *arXiv preprint arXiv:2511.14759*.

Amrani, Elad, Rami Ben-Ari, Daniel Rotman, and Alex M. Bronstein (2021). Noise Estimation Using Density Estimation for Self-Supervised Multimodal Learning. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, the Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, pp. 6644–6652. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16822.

Andreas, Jacob, Dan Klein, and Sergey Levine (2017). Modular Multitask Reinforcement Learning With Policy Sketches. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 166–175. URL: http://proceedings.mlr.press/v70/andreas17a.html.

Andrychowicz, OpenAI: Marcin, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. (2020). Learning Dexterous in-Hand Manipulation. In: *The International Journal of Robotics Research* 39.1, pp. 3–20.

Appalaraju, Srikar, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha (2021). DocFormer: End-to-End Transformer for Document Understanding. In: *arXiv preprint arXiv: Arxiv-2106.11539*.

Arulkumaran, Kai, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath (2017). Deep Reinforcement Learning: A Brief Survey. In: *IEEE Signal Processing Magazine* 34.6, pp. 26–38.

Austin, Jacob, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton (2021). Program Synthesis With Large Language Models. In: *arXiv preprint arXiv: Arxiv-2108.07732*.

AutoGPT (2023). Significant-Gravitas/Auto-Gpt: An Experimental Open-Source Attempt to Make GPT-4 Fully Autonomous. URL: https://github.com/Significant-Gravitas/Auto-GPT/tree/master.

Baker, Bowen, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune (2022). Video Pretraining (Vpt): Learning to Act by Watching Unlabeled Online Videos. In: *Advances in Neural Information Processing Systems* 35, pp. 24639–24654.

Batra, Dhruv, Angel X. Chang, Sonia Chernova, Andrew J. Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, Manolis Savva, and Hao Su (2020). Rearrangement: A Challenge for Embodied AI. In: *arXiv preprint arXiv: Arxiv-2011.01975*.

Bengio, Yoshua, Jérôme Louradour, Ronan Collobert, and Jason Weston (2009). Curriculum Learning. In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48.

Berner, Christopher, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. (2019). Dota 2 With Large Scale Deep Reinforcement Learning. In: *arXiv preprint arXiv:1912.06680*.

Bharadhwaj, Homanga, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani (2024a). Gen2act: Human Video Generation in Novel Scenarios Enables Generalizable Robot Manipulation. In: *arXiv preprint arXiv:2409.16283*.

Bharadhwaj, Homanga, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani (2024b). Track2act: Predicting Point Tracks From Internet Videos Enables Diverse Zero-Shot Robot Manipulation. In: *arXiv E-Prints*, arXiv–2405.

Bharadhwaj, Homanga, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar (2024c). RoboAgent: Generalization and Efficiency in Robot Manipulation via Semantic Augmentations and Action Chunking. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*.

Black, Kevin et al. (2024). Π_0: A Vision-Language-Action Flow Model for General Robot Control. arXiv: 2410.24164 [cs.LG]. URL: https://arxiv.org/abs/2410.24164.

Blukis, Valts, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi (2021). A Persistent Spatial Semantic Representation for High-Level Natural Language Instruction Execution. In: *5th Annual Conference on Robot Learning*. URL: https://openreview.net/forum?id=NeGDZeyjcKa.

Bommasani, Rishi, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. (2021). On the Opportunities and Risks of Foundation Models. In: *arXiv preprint arXiv:2108.07258*.

Booth, Serena, W Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, and Alessandro Allievi (2023). The Perils of Trial-and-Error Reward Design: Misdesign Through Overfitting and Invalid Task Specifications. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 5, pp. 5920–5929.

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). OpenAI Gym. In: *arXiv preprint arXiv: Arxiv-1606.01540*.

Brohan, Anthony, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. (2023a). Do as I Can, Not as I Say: Grounding Language in Robotic Affordances. In: *Conference on Robot Learning*. PMLR, pp. 287–318.

Brohan, Anthony et al. (2022). RT-1: Robotics Transformer for Real-World Control at Scale. In: *arXiv preprint arXiv:2212.06817*.

Brohan, Anthony et al. (2023b). RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In: *arXiv preprint arXiv:2307.15818*.

Brooks, Tim, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. (2024). Video Generation Models as World Simulators. 2024. In: *URL Https://openai. Com/Research/Video-Generation-Models-as-World-Simulators* 3, p. 1.

Brown, Tom B. et al. (2020). Language Models Are Few-Shot Learners. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Bruce, Jake, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. (2024). Genie: Generative Interactive Environments. In: *Forty-first International Conference on Machine Learning*.

Brunke, Lukas, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig (2021). Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning. In: *arXiv preprint arXiv: Arxiv-2108.06266*.

Bu, Qingwen, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. (2025). Agibot World Colosseo: A Large-Scale Manipulation Platform for Scalable and Intelligent Embodied Systems. In: *arXiv preprint arXiv:2503.06669*.

Bubeck, Sébastien, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang (2023). Sparks of Artificial General Intelligence: Early Experiments With GPT-4. In: *arXiv preprint arXiv: Arxiv-2303.12712*.

Cai, Shaofei, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang (2023). Open-World Multi-Task Control Through Goal-Aware Representation Learning and Adaptive Horizon Prediction. In: *arXiv preprint arXiv: Arxiv-2301.10034*.

Cao, Tianshi, Jingkang Wang, Yining Zhang, and Sivabalan Manivasagam (2020). BabyAI++: Towards Grounded-Language Learning Beyond Memorization. In: *arXiv preprint arXiv: Arxiv-2004.07200*.

Introducing chatgpt (2022). URL: https://openai.com/blog/chatgpt.

Chatzilygeroudis, Konstantinos, Vassilis Vassiliades, and Jean-Baptiste Mouret (2018). Reset-Free Trial-and-Error Learning for Robot Damage Recovery. In: *Robotics and Autonomous Systems* 100, pp. 236–250.

Cheang, Chi-Lam, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu (2024). GR-2: A Generative Video-Language-Action Model With Web-Scale Knowledge for Robot Manipulation. In: *arXiv preprint arXiv:2410.06158*.

Chen, Annie S., Suraj Nair, and Chelsea Finn (2021a). Learning Generalizable Robotic Reward Functions From in-the-Wild Human Videos. In: *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*. Ed. by Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh. DOI: 10.15607/RSS.2021.XVII.012. URL: https://doi.org/10.15607/RSS.2021.XVII.012.

Chen, Lili, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch (2021b). Decision Transformer: Reinforcement Learning via Sequence Modeling. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, Virtual*. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, pp. 15084–15097. URL: https://proceedings.neurips.cc/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

Chen, Mark, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. (2021c). Evaluating Large Language Models Trained on Code. In: *arXiv preprint arXiv:2107.03374*.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton (2020). A Simple Framework for Contrastive Learning of Visual Representations. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 1597–1607. URL: http://proceedings.mlr.press/v119/chen20j.html.

Chen, Ting, Saurabh Saxena, Lala Li, David J. Fleet, and Geoffrey E. Hinton (2022a). Pix2seq: A Language Modeling Framework for Object Detection. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. URL: https://openreview.net/forum?id=e42KbIw6Wb.

Chen, Ting, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J. Fleet, and Geoffrey Hinton (2022b). A Unified Sequence Interface for Vision Tasks. In: *arXiv preprint arXiv: Arxiv-2206.07669*.

Chen, Xinyun, Chang Liu, and Dawn Song (2019). Execution-Guided Neural Program Synthesis. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=H1gfOiAqYm.

Chen, Xinyun, Dawn Song, and Yuandong Tian (2021d). Latent Execution for Neural Program Synthesis. In: *arXiv preprint arXiv: Arxiv-2107.00101*.

Chen, Yuanpei, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang (2022c). Towards Human-Level Bimanual Dexterous Manipulation With Reinforcement Learning. In: *Advances in Neural Information Processing Systems* 35, pp. 5150–5163.

Chen, Zoey, Sho Kiami, Abhishek Gupta, and Vikash Kumar (2023). GenAug: Retargeting Behaviors to Unseen Situations via Generative Augmentation. In: *arXiv preprint arXiv:2302.06671*.

Cheng, Xuxin, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang (2024). Expressive Whole-Body Control for Humanoid Robots. In: *arXiv preprint arXiv:2402.16796*.

Chevalier-Boisvert, Maxime, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio (2019). BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=rJeXCo0cYX.

Chi, Cheng, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song (2024a). Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In: *The International Journal of Robotics Research*.

Chi, Cheng, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song (2024b). Universal Manipulation Interface: In-the-Wild Robot Teaching Without in-the-Wild Robots. In: *Proceedings of Robotics: Science and Systems (RSS)*.

Chiang, Hao-Tien Lewis, Aleksandra Faust, Marek Fiser, and Anthony Francis (2019). Learning Navigation Behaviors End-to-End With Autorl. In: *IEEE Robotics and Automation Letters* 4.2, pp. 2007–2014.

Cho, Jaemin, Jie Lei, Hao Tan, and Mohit Bansal (2021). Unifying Vision-and-Language Tasks via Text Generation. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 1931–1942. URL: http://proceedings.mlr.press/v139/cho21a.html.

Chowdhery, Aakanksha et al. (2022). PaLM: Scaling Language Modeling With Pathways. In: *arXiv preprint arXiv: Arxiv-2204.02311*.

Christiano, Paul F, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei (2017). Deep Reinforcement Learning From Human Preferences. In: *Advances in Neural Information Processing Systems* 30.

Chung, Hyung Won et al. (2022). Scaling Instruction-Finetuned Language Models. In: *arXiv preprint arXiv: Arxiv-2210.11416*.

Cobbe, Karl, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman (2021). Training Verifiers to Solve Math Word Problems. In: *arXiv preprint arXiv: Arxiv-2110.14168*.

Collins, Jack, Shelvin Chand, Anthony Vanderkop, and David Howard (2021). A Review of Physics Simulators for Robotic Applications. In: *IEEE Access* 9, pp. 51416–51431.

Common crawl (2012). https://commoncrawl.org/. Accessed: 2022-06-06.

Conti, Edoardo, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth O. Stanley, and Jeff Clune (2018). Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents. In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, pp. 5032–5043. URL: https://proceedings.neurips.cc/paper/2018/hash/b1301141feffabac455e1f90a7de2054-Abstract.html.

Dalal, Murtaza, Ajay Mandlekar, Caelan Reed Garrett, Ankur Handa, Ruslan Salakhutdinov, and Dieter Fox (2023). Imitating Task and Motion Planning With Visuomotor Transformers. In: *Conference on Robot Learning*. PMLR, pp. 2565–2593.

Damen, Dima, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. (2018). Scaling Egocentric Vision: The Epic-Kitchens Dataset. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 720–736.

Dasari, Sudeep, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn (2019). RoboNet: Large-Scale Multi-Robot Learning. In: *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, pp. 885–897. URL: http://proceedings.mlr.press/v100/dasari20a.html.

Dasari, Sudeep and Abhinav Gupta (2020). Transformers for One-Shot Visual Imitation. In: *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*. Ed. by Jens Kober, Fabio Ramos, and Claire J. Tomlin. Vol. 155. Proceedings of Machine Learning Research. PMLR, pp. 2071–2084. URL: https://proceedings.mlr.press/v155/dasari21a.html.

Dass, Shivin, Wensi Ai, Yuqian Jiang, Samik Singh, Jiaheng Hu, Ruohan Zhang, Peter Stone, Ben Abbatematteo, and Roberto Martín-Martín (2024). Telemoma: A Modular and Versatile Teleoperation System for Mobile Manipulation. In: *arXiv preprint arXiv:2403.07869*.

Dennis, Michael, Natasha Jaques, Eugene Vinitsky, Alexandre M. Bayen, Stuart Russell, Andrew Critch, and Sergey Levine (2020). Emergent Complexity and Zero-Shot Transfer via Unsupervised Environment Design. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. URL: https://proceedings.neurips.cc/paper/2020/hash/%20985e9a46e10005356bbaf194249f6856-Abstract.html.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

Ding, Yan, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang (2023). Task and Motion Planning With Large Language Models for Object Rearrangement. In: *arXiv preprint arXiv:2303.06247*.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2021). An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

Driess, Danny et al. (2023). PaLM-E: An Embodied Multimodal Language Model. In: *arXiv preprint arXiv:2303.03378*.

Du, Yuqing, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi (2023a). Vision-Language Models as Success Detectors. In: *arXiv preprint arXiv:2303.07280*.

Du, Yuqing, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas (2023b). Guiding Pretraining in Reinforcement Learning With Large Language Models. In: *arXiv preprint arXiv:2302.06692*.

Duan, Jiafei, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan (2022). A Survey of Embodied AI: From Simulators to Research Tasks. In: *IEEE Trans. Emerg. Top. Comput. Intell.* 6.2, pp. 230–244. DOI: 10.1109/TETCI.2022.3141105. URL: https://doi.org/10.1109/TETCI.2022.3141105.

Duan, Yan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba (2017). One-Shot Imitation Learning. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett,

pp. 1087–1098. URL: https://proceedings.neurips.cc/paper/2017/hash/ba3866600c3540f67c1e9575e213be0a-Abstract.html.

Dulac-Arnold, Gabriel, Daniel Mankowitz, and Todd Hester (2019). Challenges of Real-World Reinforcement Learning. In: *arXiv preprint arXiv:1904.12901*.

Ebert, Frederik, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine (June 2022). Bridge Data: Boosting Generalization of Robotic Skills With Cross-Domain Datasets. In: *Proceedings of Robotics: Science and Systems*. New York City, NY, USA. DOI: 10.15607/RSS.2022.XVIII.063.

Ecoffet, Adrien, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune (2019). Go-Explore: A New Approach for Hard-Exploration Problems. In: *arXiv preprint arXiv: Arxiv-1901.10995*.

Edwards, Ashley, Himanshu Sahni, Yannick Schroecker, and Charles Isbell (2019). Imitating Latent Policies From Observation. In: *International Conference on Machine Learning*. PMLR, pp. 1755–1763.

Ellis, Kevin, Maxwell I. Nye, Yewen Pu, Felix Sosa, Josh Tenenbaum, and Armando Solar-Lezama (2019). Write, Execute, Assess: Program Synthesis With a REPL. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 9165–9174. URL: https://proceedings.neurips.cc/paper/2019/hash/50d2d2262762648589b1943078712aa6-Abstract.html.

Ellis, Kevin, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B. Tenenbaum (2020). DreamCoder: Growing Generalizable, Interpretable Knowledge With Wake-Sleep Bayesian Program Learning. In: *arXiv preprint arXiv: Arxiv-2006.08381*.

New and improved embedding model (2022). URL: https://openai.com/blog/new-and-improved-embedding-model.

Eysenbach, Benjamin, Abhishek Gupta, Julian Ibarz, and Sergey Levine (2019). Diversity Is All You Need: Learning Skills Without a Reward Function. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=SJx63jRqFm.

Fan, Linxi, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Animashree Anandkumar (2021). SECANT: Self-Expert Cloning for Zero-Shot Generalization of Visual Policies. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 3088–3099. URL: http://proceedings.mlr.press/v139/fan21c.html.

Fan, Linxi, Guanzhi Wang*, Yunfan Jiang*, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar (2022). Minedojo: Building Open-Ended Embodied Agents With Internet-Scale Knowledge. In: *Advances in Neural Information Processing Systems* 35, pp. 18343–18362. URL: https://papers.nips.cc/paper_files/paper/2022/file/74a67268c5cc5910f64938cac4526a90-Paper-Datasets_and_Benchmarks.pdf.

Fang, Hao-Shu, Hongjie Fang, Zhenyu Tang, Jirong Liu, Junbo Wang, Haoyi Zhu, and Cewu Lu (2023). RH20T: A Robotic Dataset for Learning Diverse Skills in One-Shot. In: *RSS 2023 Workshop on Learning for Task and Motion Planning*.

Fang, Hongjie, Hao-Shu Fang, Yiming Wang, Jieji Ren, Jingjing Chen, Ruo Zhang, Weiming Wang, and Cewu Lu (2024). AirExo: Low-Cost Exoskeletons for Learning Whole-Arm Manipulation in the Wild. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 15031–15038.

Farhang, Alexander R, Brendan Mulcahy, Daniel Holden, Iain Matthews, and Yisong Yue (2024). Humanlike Behavior in a Third-Person Shooter With Imitation Learning. In: *2024 IEEE Conference on Games (CoG)*. IEEE, pp. 1–4.

Faust, Aleksandra, Anthony Francis, and Dar Mehta (2019). Evolving Rewards to Automate Reinforcement Learning. In: *arXiv preprint arXiv:1905.07628*.

Finn, Chelsea, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine (2017). One-Shot Visual Imitation Learning via Meta-Learning. In: *arXiv preprint arXiv: Arxiv-1709.04905*.

Floreano, Dario and Robert J Wood (2015). Science, Technology and the Future of Small Autonomous Drones. In: *Nature* 521.7553, pp. 460–466.

Forestier, Sébastien, Rémy Portelas, Yoan Mollard, and Pierre-Yves Oudeyer (2022). Intrinsically Motivated Goal Exploration Processes With Automatic Curriculum Learning. In: *The Journal of Machine Learning Research* 23.1, pp. 6818–6858.

Fu, Justin, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama (2019). From Language to Goals: Inverse Reinforcement Learning for Vision-Based Instruction Following. In: *arXiv preprint arXiv:1902.07742*.

Fu, Zipeng, Tony Z Zhao, and Chelsea Finn (2024). Mobile Aloha: Learning Bimanual Mobile Manipulation With Low-Cost Whole-Body Teleoperation. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*.

Fuchs, Florian, Yunlong Song, Elia Kaufmann, Davide Scaramuzza, and Peter Dürr (2021). Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning. In: *IEEE Robotics Autom. Lett.* 6.3, pp. 4257–4264. DOI: 10.1109/LRA.2021.3064284. URL: https://doi.org/10.1109/LRA.2021.3064284.

Fujimoto, Scott, Herke Hoof, and David Meger (2018). Addressing Function Approximation Error in Actor-Critic Methods. In: *International Conference on Machine Learning*. PMLR, pp. 1587–1596.

Gao, Leo, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. (2020). The Pile: An 800gb Dataset of Diverse Text for Language Modeling. In: *arXiv preprint arXiv:2101.00027*.

Gao, Peng, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao (2021). CLIP-Adapter: Better Vision-Language Models With Feature Adapters. In: *arXiv preprint arXiv: Arxiv-2110.04544*.

Garcıa, Javier and Fernando Fernández (2015). A Comprehensive Survey on Safe Reinforcement Learning. In: *Journal of Machine Learning Research* 16.1, pp. 1437–1480.

Garrett, Caelan, Ajay Mandlekar, Bowen Wen, and Dieter Fox (2024). SkillMimicGen: Automated Demonstration Generation for Efficient Skill Learning and Deployment. In: *arXiv preprint arXiv:2410.18907*.

Gerblick, Jordan (Dec. 2021). Minecraft, the Most-Watched Game on YouTube, Passes 1 Trillion Views. URL: https://www.gamesradar.com/minecraft-the-most-watched-game-%20on-youtube-passes-1-trillion-views/.

Ghiasi, Golnaz, Barret Zoph, Ekin D. Cubuk, Quoc V. Le, and Tsung-Yi Lin (2021). Multi-Task Self-Training for Learning General Representations. In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, pp. 8836–8845. DOI: 10.1109/ICCV48922.2021.00873. URL: https://doi.org/10.1109/ICCV48922.2021.00873.

Goyal, Raghav, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. (2017). The" Something Something" Video Database for Learning and Evaluating Visual Common Sense. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5842–5850.

Grauman, Kristen, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. (2022). Ego4d: Around the World in 3,000 Hours of Egocentric Video. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18995–19012.

Grauman, Kristen, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. (2024). Ego-Exo4d: Understanding Skilled Human Activity From First-and Third-Person Perspectives. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19383–19400.

Gray, Jonathan, Kavya Srinet, Yacine Jernite, Haonan Yu, Zhuoyuan Chen, Demi Guo, Siddharth Goyal, C. Lawrence Zitnick, and Arthur Szlam (2019). CraftAssist: A Framework for Dialogue-Enabled Interactive Agents. In: *arXiv preprint arXiv: Arxiv-1907.08584*.

Grbic, Djordje, Rasmus Berg Palm, Elias Najarro, Claire Glanois, and Sebastian Risi (2021). EvoCraft: A New Challenge for Open-Endedness. In: Springer International Publishing, pp. 325–340. DOI: 10.1007/978-3-030-72699-7_21. URL: http://link.springer.com/content/pdf/10.1007/978-3-030-%2072699-7_21.

Gu, Jiayuan, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. (2023). ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills. In: *The Eleventh International Conference on Learning Representations*.

Guo, Yanjiang, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn (2025). CTRL-World: A Controllable Generative World Model for Robot Manipulation. In: *arXiv preprint arXiv:2510.10125*.

Gupta, Abhishek, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman (2019). Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning. In: *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, pp. 1025–1037. URL: http://proceedings.mlr.press/v100/gupta20a.html.

Gupta, Abhishek, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine (2021). Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors Without Human Intervention. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6664–6671.

Guss, William H, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov (2019a). Minerl: A Large-Scale Dataset of Minecraft Demonstrations. In: *arXiv preprint arXiv:1907.13440*.

Guss, William H., Mario Ynocente Castro, Sam Devlin, Brandon Houghton, Noboru Sean Kuno, Crissman Loomis, Stephanie Milani, Sharada Mohanty, Keisuke Nakata, Ruslan Salakhutdinov, John Schulman, Shinya Shiroshita, Nicholay Topin, Avinash Ummadisingu, and Oriol Vinyals (2021). The MineRL 2020 Competition on Sample Efficient Reinforcement Learning Using Human Priors. In: *arXiv preprint arXiv: Arxiv-2101.11071*.

Guss, William H., Cayden Codel, Katja Hofmann, Brandon Houghton, Noboru Kuno, Stephanie Milani, Sharada Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, Manuela Veloso, and Phillip Wang (2019b). The MineRL 2019

Competition on Sample Efficient Reinforcement Learning Using Human Priors. In: *arXiv preprint arXiv: Arxiv-1904.10079*.

Ha, Huy, Pete Florence, and Shuran Song (2023). Scaling Up and Distilling Down: Language-Guided Robot Skill Acquisition. In: *Conference on Robot Learning*. PMLR, pp. 3766–3777.

Haarnoja, Tuomas, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine (2018). Soft Actor-Critic Algorithms and Applications. In: *arXiv preprint arXiv: Arxiv-1812.05905*.

Hadfield-Menell, Dylan, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan (2017). Inverse Reward Design. In: *Advances in Neural Information Processing Systems* 30.

Hafner, Danijar (2021). Benchmarking the Spectrum of Agent Capabilities. In: *arXiv preprint arXiv: Arxiv-2109.06780*.

Hafner, Danijar, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap (2023). Mastering Diverse Domains Through World Models. In: *arXiv preprint arXiv:2301.04104*.

Hafner, Danijar, Wilson Yan, and Timothy Lillicrap (2025). Training Agents Inside of Scalable World Models. arXiv: 2509.24527 [cs.AI]. URL: https://arxiv.org/abs/2509.24527.

Handa, Ankur, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. (2023). Dextreme: Transfer of Agile in-Hand Manipulation From Simulation to Reality. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 5977–5984.

Hanu, Laura and Unitary team (2020). Detoxify. Github. https://github.com/unitaryai/detoxify.

He, Kaiming, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick (2020). Momentum Contrast for Unsupervised Visual Representation Learning. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, pp. 9726–9735. DOI: 10.1109/CVPR42600.2020.00975. URL: https://doi.org/10.1109/CVPR42600.2020.00975.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick (2017). Mask R-CNN. In: *arXiv preprint arXiv: Arxiv-1703.06870*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (Dec. 2015). Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs]. DOI: 10.48550/arXiv.1512.03385. URL: http://arxiv.org/abs/1512.03385 (visited on 09/28/2022).

Henderson, Matthew, Paweł Budzianowski, Iñigo Casanueva, Sam Coope, Daniela Gerz, Girish Kumar, Nikola Mrkšić, Georgios Spithourakis, Pei-Hao Su, Ivan Vulić, and Tsung-Hsien Wen (2019). A Repository of Conversational Datasets. In: *arXiv preprint arXiv: Arxiv-1904.06472*.

Henderson, Peter, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger (2018). Deep Reinforcement Learning That Matters. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1.

Hermann, Karl Moritz, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom (2017). Grounded Language Learning in a Simulated 3D World. In: *arXiv preprint arXiv: Arxiv-1706.06551*.

Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 6626–6637. URL: https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html.

Ho, Jonathan and Stefano Ermon (2016). Generative Adversarial Imitation Learning. In: *Advances in Neural Information Processing Systems* 29.

Hu, Xixi, Bo Liu, Xingchao Liu, and Qiang Liu (2024). AdaFlow: Imitation Learning With Variance-Adaptive Flow-Based Policies. In: *arXiv preprint arXiv:2402.04292*.

Huang, Jiangyong, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang (2024). An Embodied Generalist Agent in 3D World. In: *Proceedings of the International Conference on Machine Learning (ICML)*.

Huang, Siyuan, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li (2023a). Instruct2Act: Mapping Multi-Modality Instructions to Robotic Actions With Large Language Model. In: *arXiv preprint arXiv:2305.11176*.

Huang, Wenlong, Pieter Abbeel, Deepak Pathak, and Igor Mordatch (2022a). Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In: *International Conference on Machine Learning*. PMLR, pp. 9118–9147.

Huang, Wenlong, Fei Xia, Dhruv Shah, Danny Driess, Andy Zeng, Yao Lu, Pete Florence, Igor Mordatch, Sergey Levine, Karol Hausman, et al. (2023b). Grounded Decoding: Guiding Text Generation With Grounded Models for Embodied Agents. In: *Advances in Neural Information Processing Systems* 36, pp. 59636–59661.

Huang, Wenlong, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. (2022b). Inner Monologue: Embodied Reasoning Through Planning With Language Models. In: *arXiv preprint arXiv:2207.05608*.

Huizinga, Joost and Jeff Clune (2022). Evolving Multimodal Robot Behavior via Many Stepping Stones With the Combinatorial Multiobjective Evolutionary Algorithm. In: *Evolutionary Computation* 30.2, pp. 131–164.

Iyer, Aadhithya, Zhuoran Peng, Yinlong Dai, Irmak Guzey, Siddhant Haldar, Soumith Chintala, and Lerrel Pinto (2024). Open Teach: A Versatile Teleoperation System for Robotic Manipulation. In: *arXiv preprint arXiv:2403.07870*.

Jaegle, Andrew, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira (2021a). Perceiver IO: A General Architecture for Structured Inputs & Outputs. In: *arXiv preprint arXiv: Arxiv-2107.14795*.

Jaegle, Andrew, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira (2021b). Perceiver: General Perception With Iterative Attention. In: *arXiv preprint arXiv: Arxiv-2103.03206*.

James, Stephen, Zicong Ma, David Rovick Arrojo, and Andrew J Davison (2020). Rlbench: The Robot Learning Benchmark & Learning Environment. In: *IEEE Robotics and Automation Letters* 5.2, pp. 3019–3026.

Janner, Michael, Qiyang Li, and Sergey Levine (2021). Offline Reinforcement Learning as One Big Sequence Modeling Problem. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, Virtual*. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, pp. 1273–1286. URL: https://proceedings.neurips.cc/paper/2021/hash/099fe6b0b444c23836c4a5d07346082b-Abstract.html.

Jiang, Yunfan, Agrim Gupta*, Zichen Zhang*, Guanzhi Wang*, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan (July 2023). VIMA: Robot Manipulation With Multimodal Prompts. In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 14975–15022. URL: https://proceedings.mlr.press/v202/jiang23b.html.

Jiang, Zhenyu, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu (2024). DexMimicGen: Automated Data Generation for Bimanual Dexterous Manipulation via Imitation Learning. In.

Johnson, Matthew, Katja Hofmann, Tim Hutton, and David Bignell (2016). The Malmo Platform for Artificial Intelligence Experimentation. In: *International Joint Conference on Artificial Intelligence*. URL: https://dl.acm.org/doi/10.5555/3061053.3061259.

Juliani, Arthur, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange (2019). Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, pp. 2684–2691. DOI: 10.24963/ijcai.2019/373. URL: https://doi.org/10.24963/ijcai.2019/373.

Kahneman, Daniel (2011). Thinking, Fast and Slow.

Kanervisto, Anssi et al. (2022). MineRL Diamond 2021 Competition: Overview, Results, and Lessons Learned. In: *arXiv preprint arXiv: Arxiv-2202.10583*.

Kanitscheider, Ingmar, Joost Huizinga, David Farhi, William Hebgen Guss, Brandon Houghton, Raul Sampedro, Peter Zhokhov, Bowen Baker, Adrien Ecoffet, Jie Tang, Oleg Klimov, and Jeff Clune (2021). Multi-Task Curriculum Learning in a Complex, Visual, Hard-Exploration Domain: Minecraft. In: *arXiv preprint arXiv: Arxiv-2106.14876*.

Karamcheti, Siddharth, Suraj Nair, Annie S Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang (2023). Language-Driven Representation Learning for Robotics. In: *arXiv preprint arXiv:2302.12766*.

Kareer, Simar, Dhruv Patel, Ryan Punamiya, Pranay Mathur, Shuo Cheng, Chen Wang, Judy Hoffman, and Danfei Xu (2024). EgoMimic: Scaling Imitation Learning via Egocentric Video. arXiv: 2410.24221.

Kay, Will, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman (2017). The Kinetics Human Action Video Dataset. In: *arXiv preprint arXiv: Arxiv-1705.06950*.

Khan, Salman, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah (2021). Transformers in Vision: A Survey. In: *arXiv preprint arXiv: Arxiv-2101.01169*.

Khazatsky, Alexander et al. (2024). DROID: A Large-Scale in-the-Wild Robot Manipulation Dataset. In: *arXiv preprint arXiv:2403.12945*.

Kim, Byeongchang, Hyunwoo Kim, and Gunhee Kim (2019). Abstractive Summarization of Reddit Posts With Multi-Level Memory Networks. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Com-

putational Linguistics, pp. 2519–2531. DOI: 10.18653/v1/n19-1260. URL: https://doi.org/10.18653/v1/n19-1260.

Kim, Moo Jin, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. (2024). Openvla: An Open-Source Vision-Language-Action Model. In: *arXiv preprint arXiv:2406.09246*.

Kingma, Diederik P. and Jimmy Ba (2017). Adam: A Method for Stochastic Optimization. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

Kiseleva, Julia, Ziming Li, Mohammad Aliannejadi, Shrestha Mohanty, Maartje ter Hoeve, Mikhail Burtsev, Alexey Skrynnik, Artem Zholus, Aleksandr Panov, Kavya Srinet, Arthur Szlam, Yuxuan Sun, Katja Hofmann, Michel Galley, and Ahmed Awadallah (2021). NeurIPS 2021 Competition IGLU: Interactive Grounded Language Understanding in a Collaborative Environment. In: *arXiv preprint arXiv: Arxiv-2110.06536*.

Knox, W Bradley, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone (2023). Reward (Mis) Design for Autonomous Driving. In: *Artificial Intelligence* 316, p. 103829.

Kober, Jens, J Andrew Bagnell, and Jan Peters (2013). Reinforcement Learning in Robotics: A Survey. In: *The International Journal of Robotics Research* 32.11, pp. 1238–1274.

Kolesnikov, Alexander, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby (2020). Big Transfer (BiT): General Visual Representation Learning. In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part V*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Vol. 12350. Lecture Notes in Computer Science. Springer, pp. 491–507. DOI: 10.1007/978-3-030-58558-7_29. URL: https://doi.org/10.1007/978-3-030-58558-7%5C_29.

Kolve, Eric, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi (2017). AI2-THOR: An Interactive 3D Environment for Visual AI. In: *arXiv preprint arXiv: Arxiv-1712.05474*.

Kostrikov, Ilya, Denis Yarats, and Rob Fergus (2020). Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning From Pixels. In: *arXiv preprint arXiv: Arxiv-2004.13649*.

Küttler, Heinrich, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel (2020). The NetHack Learning Environment. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 7671–7684. URL: https://proceedings.neurips.

cc/paper/2020/file/%20569ff987c643b4bedf504efda8f786c2-Paper.pdf.

Kwon, Minae, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh (2023). Reward Design With Language Models. In: *arXiv preprint arXiv:2303.00001*.

Langdon, WB (2005). Pfeiffer–a Distributed Open-Ended Evolutionary System. In: *Aisb*. Vol. 5. Citeseer, pp. 7–13.

Le, Hung, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C. H. Hoi (2022). CodeRL: Mastering Code Generation Through Pretrained Models and Deep Reinforcement Learning. In: *arXiv preprint arXiv: Arxiv-2207.01780*.

Li, Chengshu, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, C. Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese (2021). iGibson 2.0: Object-Centric Simulation for Robot Learning of Everyday Household Tasks. In: *Conference on Robot Learning, 8-11 November 2021, London, UK*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, pp. 455–465. URL: https://proceedings.mlr.press/v164/li22b.html.

Li, Shuang, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, Jacob Andreas, Igor Mordatch, Antonio Torralba, and Yuke Zhu (2022a). Pre-Trained Language Models for Interactive Decision-Making. In: *arXiv preprint arXiv: Arxiv-2202.01771*.

Li, Xinghang, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. (2023). Vision-Language Foundation Models as Effective Robot Imitators. In: *arXiv preprint arXiv:2311.01378*.

Li, Yujia et al. (2022b). Competition-Level Code Generation With AlphaCode. In: *arXiv preprint arXiv: Arxiv-2203.07814*.

Li, Zhiqi, Guo Chen, Shilong Liu, Shihao Wang, Vibashan VS, Yishen Ji, Shiyi Lan, Hao Zhang, Yilin Zhao, Subhashree Radhakrishnan, et al. (2025). Eagle 2: Building Post-Training Data Strategies From Scratch for Frontier Vision-Language Models. In: *arXiv preprint arXiv:2501.14818*.

Liang, Jacky, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng (2023). Code as Policies: Language Model Programs for Embodied Control. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 9493–9500.

Liang, Paul Pu, Yiwei Lyu, Xiang Fan, Shentong Mo, Dani Yogatama, Louis-Philippe Morency, and Ruslan Salakhutdinov (2022). HighMMT: Towards Modality and Task Generalization for High-Modality Representation Learning. In: *arXiv preprint arXiv: Arxiv-2203.01311*.

Liang, William, Sam Wang, Hung-Ju Wang, Osbert Bastani, Dinesh Jayaraman, and Yecheng Jason Ma (2024). Eurekaverse: Environment Curriculum Generation via Large Language Models. In: *arXiv preprint arXiv:2411.01775*.

Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (2016). Continuous Control With Deep Reinforcement Learning. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1509.02971.

Lim, Michael H., Andy Zeng, Brian Ichter, Maryam Bandari, Erwin Coumans, Claire Tomlin, Stefan Schaal, and Aleksandra Faust (2021). Multi-Task Learning With Sequence-Conditioned Transporter Networks. In: *arXiv preprint arXiv: Arxiv-2109.07578*.

Lin, Kevin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg (2023). Text2Motion: From Natural Language Instructions to Feasible Plans. In: *Autonomous Robots*.

Lin, Tianyang, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu (2021a). A Survey of Transformers. In: *arXiv preprint arXiv: Arxiv-2106.04554*.

Lin, Zichuan, Junyou Li, Jianing Shi, Deheng Ye, Qiang Fu, and Wei Yang (2021b). JueWu-MC: Playing Minecraft With Sample-Efficient Hierarchical Reinforcement Learning. In: *arXiv preprint arXiv: Arxiv-2112.04907*.

Lin, Zongyu, Wei Liu, Chen Chen, Jiasen Lu, Wenze Hu, Tsu-Jui Fu, Jesse Allardice, Zhengfeng Lai, Liangchen Song, Bowen Zhang, et al. (2024). STIV: Scalable Text and Image Conditioned Video Generation. In: *arXiv preprint arXiv:2412.07730*.

Lipman, Yaron, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le (2022). Flow Matching for Generative Modeling. In: *arXiv preprint arXiv:2210.02747*.

Liu, Bo, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone (2023a). Llm+ P: Empowering Large Language Models With Optimal Planning Proficiency. In: *arXiv preprint arXiv:2304.11477*.

Liu, Shikun, Linxi Fan, Edward Johns, Zhiding Yu, Chaowei Xiao, and Anima Anandkumar (2023b). Prismer: A Vision-Language Model With an Ensemble of Experts. In: *arXiv preprint arXiv: Arxiv-2303.02506*.

Liu, Xingchao, Chengyue Gong, and Qiang Liu (2022a). Flow Straight and Fast: Learning to Generate and Transfer Data With Rectified Flow. In: *arXiv preprint arXiv:2209.03003*.

Liu, Yiheng, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, and Bao Ge (2023c). Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models. In: *arXiv preprint arXiv: Arxiv-2304.01852*.

Liu, Yunze, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi (June 2022b). HOI4D: A 4D Egocentric Dataset for Category-Level Human-Object Interaction. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21013–21022.

Loshchilov, Ilya and Frank Hutter (2016). SGDR: Stochastic Gradient Descent With Warm Restarts. In: *arXiv preprint arXiv: Arxiv-1608.03983*.

– (2019). Decoupled Weight Decay Regularization. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

Lowe, David G (2004). Distinctive Image Features From Scale-Invariant Keypoints. In: *International Journal of Computer Vision* 60, pp. 91–110.

Lu, Jiasen, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi (2022). Unified-Io: A Unified Model for Vision, Language, and Multi-Modal Tasks. In: *arXiv preprint arXiv: Arxiv-2206.08916*.

Luo, Huaishao, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li (2021). CLIP4Clip: An Empirical Study of CLIP for End to End Video Clip Retrieval. In: *arXiv preprint arXiv: Arxiv-2104.08860*.

Luo, Zhengyi, Ye Yuan, Tingwu Wang, Chenran Li, Sirui Chen, Fernando Castañeda, Zi-Ang Cao, Jiefeng Li, David Minor, Qingwei Ben, et al. (2025). SONIC: Supersizing Motion Tracking for Natural Humanoid Whole-Body Control. In: *arXiv preprint arXiv:2511.07820*.

Lynch, Corey and Pierre Sermanet (2021). Language Conditioned Imitation Learning Over Unstructured Data. In: *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*. Ed. by Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh. DOI: 10.15607/RSS.2021.XVII.047. URL: https://doi.org/10.15607/RSS.2021.XVII.047.

Lynch, Corey, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence (2023). Interactive Language: Talking to Robots in Real Time. In: *IEEE Robotics and Automation Letters*.

Ma, Yecheng Jason, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman (2023). LIV: Language-Image Representations and Rewards for Robotic Control. In: *arXiv preprint arXiv:2306.00958*.

Ma, Yecheng Jason, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, and Anima Anandkumar (2024). Eureka: Human-Level Reward Design via Coding Large Language Models. In: *International Conference on Representation Learning* 2024. Ed. by B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, pp. 26516–26560. URL: https://proceedings.iclr.cc/paper_files/paper/2024/file/70c26937fbf3d4600b69a129031b66ec-Paper-Conference.pdf.

Ma, Yecheng Jason, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang (2022). Vip: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training. In: *arXiv preprint arXiv:2210.00030*.

Magne, Loïc*, Anas Awadalla*, Guanzhi Wang*, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, Yisong Yue, Yejin Choi, Yuke Zhu, and Linxi Fan (2026). NitroGen: An Open Foundation Model for Generalist Gaming Agents. In: *arXiv Preprint arXiv:2601.02427*. URL: https://arxiv.org/abs/2601.02427.

Makoviichuk, Denys and Viktor Makoviychuk (May 2021). Rl-Games: A High-Performance Framework for Reinforcement Learning. https://github.com/Denys88/rl_games.

Makoviychuk, Viktor, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State (2021). Isaac Gym: High Performance GPU-Based Physics Simulation for Robot Learning. In: *arXiv preprint arXiv: Arxiv-2108.10470*.

Mandi, Zhao, Homanga Bharadhwaj, Vincent Moens, Shuran Song, Aravind Rajeswaran, and Vikash Kumar (2022). CACTI: A Framework for Scalable Multi-Task Multi-Scene Visual Imitation Learning. In: *arXiv preprint arXiv:2212.05711*.

Mandlekar, Ajay, Jonathan Booher, Max Spero, Albert Tung, Anchit Gupta, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei (2019). Scaling Robot Supervision to Hundreds of Hours With Roboturk: Robotic Manipulation Dataset Through Human Reasoning and Dexterity. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1048–1055.

Mandlekar, Ajay, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox (2023). MimicGen: A Data Generation System for Scalable Robot Learning Using Human Demonstrations. In: *Conference on Robot Learning 2023*.

Mandlekar, Ajay, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese (2020). Human-in-the-Loop Imitation Learning Using Remote Teleoperation. In: *arXiv preprint arXiv:2012.06733*.

Mandlekar, Ajay, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín (2021). What Matters in Learning From Offline Human Demonstrations for Robot Manipulation. In.

Mandlekar, Ajay, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. (2018). Roboturk: A Crowdsourcing Platform for Robotic Skill Learning Through Imitation. In: *Conference on Robot Learning*. PMLR, pp. 879–893.

Mao, Hangyu, Chao Wang, Xiaotian Hao, Yihuan Mao, Yiming Lu, Chengjie Wu, Jianye Hao, Dong Li, and Pingzhong Tang (2021). SEIHAI: A Sample-Efficient Hierarchical AI for the MineRL Competition. In: *arXiv preprint arXiv: Arxiv-2111.08857*.

Miech, Antoine, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman (2020). End-to-End Learning of Visual Representations From Uncurated Instructional Videos. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, pp. 9876–9886. DOI: 10.1109/CVPR42600.2020.00990. URL: https://openaccess.thecvf.com/content%5C_CVPR%5C_2020/html/Miech%5C_End-to-End%5C_Learning%5C_of%5C_Visual%5C_Representations%5C_From%5C_Uncurated%5C_Instructional%5C_Videos%5C_CVPR%5C_2020%5C_paper.html.

Miech, Antoine, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic (2019). Howto100m: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2630–2640.

Min, So Yeon, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and R. Salakhutdinov (2021). FILM: Following Instructions in Language With Modular Methods. In: *International Conference on Learning Representations*. URL: https://arxiv.org/abs/2110.07342v3.

Minecraft wiki (2016). hhttps://minecraft.fandom.com/wiki/Minecraft_Wiki. Accessed: 2022-06-06.

PrismarineJS (2013). PrismarineJS/mineflayer: Create Minecraft Bots With a Powerful, Stable, and High Level Javascript API. URL: https://github.com/PrismarineJS/mineflayer/tree/master.

Mittal, Mayank, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. (2025). Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning. In: *arXiv preprint arXiv:2511.04831*.

Mnih, Volodymyr, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu (2016). Asynchronous Methods for Deep Reinforcement Learning. In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1928–1937. URL: http://proceedings.mlr.press/v48/mniha16.html.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013). Playing Atari With Deep Reinforcement Learning. In: *arXiv preprint arXiv:1312.5602*.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. (2015). Human-Level Control Through Deep Reinforcement Learning. In: *Nature* 518.7540, pp. 529–533.

Moritz, Philipp, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. (2018). Ray: A Distributed Framework for Emerging {AI} Applications. In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pp. 561–577.

Nachum, Ofir, Shixiang Shane Gu, Honglak Lee, and Sergey Levine (2018). Data-Efficient Hierarchical Reinforcement Learning. In: *Advances in Neural Information Processing Systems* 31.

Nair, Suraj, Eric Mitchell, Kevin Chen, Brian Ichter, Silvio Savarese, and Chelsea Finn (2021). Learning Language-Conditioned Robot Behavior From Offline Data and Crowd-Sourced Annotation. In: *Conference on Robot Learning, 8-11 November 2021, London, UK*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, pp. 1303–1315. URL: https://proceedings.mlr.press/v164/nair22a.html.

Nair, Suraj, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta (2022). R3m: A Universal Visual Representation for Robot Manipulation. In: *arXiv preprint arXiv:2203.12601*.

Nair, Varun, Elliot Schumacher, Geoffrey Tso, and Anitha Kannan (2023). DERA: Enhancing Large Language Model Completions With Dialog-Enabled Resolving Agents. In: *arXiv preprint arXiv: Arxiv-2303.17071*.

Nasiriany, Soroush, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu (2024). RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots. In: *Robotics: Science and Systems (RSS)*.

Ni, Ansong, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-tau Yih, Sida I. Wang, and Xi Victoria Lin (2023). LEVER: Learning to Verify Language-to-Code Generation With Execution. In: *arXiv preprint arXiv: Arxiv-2302.08468*.

Niekum, Scott, Andrew G. Barto, and Lee Spector (2010). Genetic Programming for Reward Function Search. In: *IEEE Transactions on Autonomous Mental Development* 2.2, pp. 83–90.

Nijkamp, Erik, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong (2022). A Conversational Paradigm for Program Synthesis. In: *arXiv preprint arXiv: Arxiv-2203.13474*.

Nottingham, Kolby, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hanna Hajishirzi, Sameer Singh, and Roy Fox (2023). Do Embodied Agents Dream of Pixelated Sheep?: Embodied Decision Making Using Language Guided World Modelling. In: *ARXIV.ORG*. DOI: 10.48550/arXiv.2301.12050.

NVIDIA (2025). OSMO Platform. Accessed: 2025-03-12. URL: https://developer.nvidia.com/osmo.

NVIDIA, Guanzhi Wang, et al. (2025). Gr00t N1: An Open Foundation Model for Generalist Humanoid Robots. In: *arXiv Preprint arXiv:2503.14734*. URL: https://arxiv.org/abs/2503.14734.

O'Connell, Michael, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung (2022). Neural-Fly Enables Rapid Learning for Agile Flight in Strong Winds. In: *Science Robotics* 7.66, eabm6597.

O'Neill, Abby, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. (2024). Open X-Embodiment: Robotic Learning Datasets and Rt-X Models: Open X-Embodiment Collaboration 0. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6892–6903.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine (2024). Octo: An Open-Source Generalist Robot Policy. In: *Proceedings of Robotics: Science and Systems*. Delft, Netherlands.

Oh, Junhyuk, Yijie Guo, Satinder Singh, and Honglak Lee (2018). Self-Imitation Learning. In: *International Conference on Machine Learning*. PMLR, pp. 3878–3887.

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals (2018). Representation Learning With Contrastive Predictive Coding. In: *arXiv preprint arXiv: Arxiv-1807.03748*.

Open X-Embodiment Collaboration et al. (2024). Open X-Embodiment: Robotic Learning Datasets and RT-X Models. International Conference on Robotics and Automation.

OpenAI (2023). GPT-4 Technical Report. arXiv: 2303.08774 [cs.CL].

OpenAI et al. (2019). Dota 2 With Large Scale Deep Reinforcement Learning. In: *arXiv preprint arXiv: Arxiv-1912.06680*.

Parisi, German Ignacio, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter (2019). Continual Lifelong Learning With Neural Networks: A Review. In: *Neural Networks* 113, pp. 54–71. DOI: 10.1016/j.neunet.2019.01.012. URL: https://doi.org/10.1016/j.neunet.2019.01.012.

Parisi, Simone, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta (2022). The Unsurprising Effectiveness of Pre-Trained Vision Models for Control. In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine

Learning Research. PMLR, pp. 17359–17371. URL: https://proceedings.mlr.press/v162/parisi22a.html.

Park, Joon Sung, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein (2023). Generative Agents: Interactive Simulacra of Human Behavior. In: *arXiv preprint arXiv: Arxiv-2304.03442*.

Parker-Holder, J, P Ball, J Bruce, V Dasagi, K Holsheimer, C Kaplanis, A Moufarek, G Scully, J Shar, J Shi, et al. (2024). Genie 2: A Large-Scale Foundation World Model. In: *URL: https://deepmind. google/discover/blog/genie-2-a-large-scale-foundation-world-model*.

Peebles, William and Saining Xie (2023). Scalable Diffusion Models With Transformers. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205.

Portelas, Rémy, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer (2020). Automatic Curriculum Learning for Deep RL: A Short Survey. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. Ed. by Christian Bessiere. ijcai.org, pp. 4819–4825. DOI: 10.24963/ijcai.2020/671. URL: https://doi.org/10.24963/ijcai.2020/671.

Potje, Guilherme, Felipe Cadar, André Araujo, Renato Martins, and Erickson R Nascimento (2024). Xfeat: Accelerated Features for Lightweight Image Matching. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2682–2691.

Puig, Xavier, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba (2018). VirtualHome: Simulating Household Activities via Programs. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, pp. 8494–8502. DOI: 10.1109/CVPR.2018.00886. URL: http://openaccess.thecvf.com/content%5C_cvpr%5C_2018/html/Puig%5C_VirtualHome%5C_Simulating%5C_Household%5C_CVPR%5C_2018%5C_paper.html.

Raad, Maria Abi, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, et al. (2024). Scaling Instructable Agents Across Many Simulated Worlds. In: *arXiv preprint arXiv:2404.10179*.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. (2021). Learning Transferable Visual Models From Natural Language Supervision. In: *International Conference on Machine Learning*. PMLR, pp. 8748–8763.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). Improving Language Understanding by Generative Pre-Training. In: *OpenAI*.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. (2019). Language Models Are Unsupervised Multitask Learners. In: *OpenAI Blog* 1.8, p. 9.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). Exploring the Limits of Transfer Learning With a Unified Text-to-Text Transformer. In: *Journal of Machine Learning Research* 21, 140:1–140:67. URL: http://jmlr.org/papers/v21/20-074.html.

Rajeswaran, Aravind, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine (2018). Learning Complex Dexterous Manipulation With Deep Reinforcement Learning and Demonstrations. arXiv: 1709.10087 [cs.LG]. URL: https://arxiv.org/abs/1709.10087.

Ramesh, Aditya, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen (2022). Hierarchical Text-Conditional Image Generation With CLIP Latents. In: *arXiv preprint arXiv: Arxiv-2204.06125*.

Ravichandar, Harish, Athanasios S Polydoros, Sonia Chernova, and Aude Billard (2020). Recent Advances in Robot Learning From Demonstration. In: *Annual Review of Control, Robotics, and Autonomous Systems* 3, pp. 297–330.

Reed, Scott, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas (2022). A Generalist Agent. arXiv: 2205.06175 [cs.AI]. URL: https://arxiv.org/abs/2205.06175.

Reid, Machel, Yutaro Yamada, and Shixiang Shane Gu (2022). Can Wikipedia Help Offline Reinforcement Learning? In: *arXiv preprint arXiv: Arxiv-2201.12122*.

Ren, Juntao, Priya Sundaresan, Dorsa Sadigh, Sanjiban Choudhury, and Jeannette Bohg (2025a). Motion Tracks: A Unified Representation for Human-Robot Transfer in Few-Shot Imitation Learning. In: *arXiv preprint arXiv:2501.06994*.

Ren, Zhongwei, Yunchao Wei, Xun Guo, Yao Zhao, Bingyi Kang, Jiashi Feng, and Xiaojie Jin (2025b). VideoWorld: Exploring Knowledge Learning From Unlabeled Videos. In: *arXiv preprint arXiv:2501.09781*.

Rozière, Baptiste, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. (2023). Code Llama: Open Foundation Models for Code. In: *arXiv preprint arXiv:2308.12950*.

Russell, Stuart J and Peter Norvig (1995). Artificial Intelligence: A Modern Approach. 1st. Englewood Cliffs, NJ, USA: Prentice Hall.

Salimans, Tim, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen (2016). Improved Techniques for Training GANs. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, pp. 2226–2234. URL: https://proceedings.neurips.cc/paper/2016/hash/8a3363abe792db2d8761d6403605aeb7-Abstract.html.

Sanh, Victor et al. (2021). Multitask Prompted Training Enables Zero-Shot Task Generalization. In: *International Conference on Learning Representations*.

Savva, Manolis, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra (Oct. 2019). Habitat: A Platform for Embodied AI Research. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Sax, Alexander, Bradley Emi, Amir R. Zamir, Leonidas Guibas, Silvio Savarese, and Jitendra Malik (2018). Mid-Level Visual Representations Improve Generalization and Sample Efficiency for Learning Visuomotor Policies. In: *arXiv preprint arXiv: Arxiv-1812.11971*.

Schick, Timo, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. In: *arXiv preprint arXiv: Arxiv-2302.04761*.

Schuhmann, Christoph, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki (2021). Laion-400m: Open Dataset of Clip-Filtered 400 Million Image-Text Pairs. In: *arXiv preprint arXiv:2111.02114*.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). Proximal Policy Optimization Algorithms. In: *arXiv preprint arXiv: Arxiv-1707.06347*.

Selenium webdriver (2011). https://www.selenium.dev/. Accessed: 2022-06-06.

Sener, F., D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao (2022). Assembly101: A Large-Scale Multi-View Video Dataset for Understanding Procedural Activities. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21096–21106.

Shah, Dhruv, Blazej Osinski, Brian Ichter, and Sergey Levine (2022). LM-Nav: Robotic Navigation With Large Pre-Trained Models of Language, Vision, and Action. In: *arXiv preprint arXiv: Arxiv-2207.04429*.

Shah, Rohin, Cody Wild, Steven H. Wang, Neel Alex, Brandon Houghton, William Guss, Sharada Mohanty, Anssi Kanervisto, Stephanie Milani, Nicholay Topin, Pieter Abbeel, Stuart Russell, and Anca Dragan (2021). The MineRL BASALT Competition on Learning From Human Feedback. In: *arXiv preprint arXiv: Arxiv-2107.01969*.

Shah, Rutav, Roberto Martín-Martín, and Yuke Zhu (2023). MUTEX: Learning Unified Policies From Multimodal Task Specifications. In: *7th Annual Conference on Robot Learning*.

Shao, Lin, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg (2021). Concept2robot: Learning Manipulation Concepts From Instructions and Human Demonstrations. In: *The International Journal of Robotics Research* 40.12-14, pp. 1419–1434.

Shen, Bokui, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne P. Tchapmi, Micael E. Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese (2020). iGibson 1.0: A Simulation Environment for Interactive Tasks in Large Realistic Scenes. In: *arXiv preprint arXiv: Arxiv-2012.02924*.

Shi, Guanya, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung (2019). Neural Lander: Stable Drone Landing Control Using Learned Dynamics. In: *2019 International Conference on Robotics and Automation (Icra)*. IEEE, pp. 9784–9790.

Shi, Tianlin Tim, Andrej Karpathy, Linxi Jim Fan, Jonathan Hernandez, and Percy Liang (2017). World of Bits: An Open-Domain Platform for Web-Based Agents. In: *Icml*. URL: https://dl.acm.org/doi/10.5555/3305890.3306005.

Shi, Wenzhe, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang (2016). Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Shinn, Noah, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao (2023). Reflexion: An Autonomous Agent With Dynamic Memory and Self-Reflection. In: *arXiv preprint arXiv:2303.11366*.

Shoeybi, Mohammad, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro (2019). Megatron-Lm: Training Multi-Billion Parameter Language Models Using Model Parallelism. In: *arXiv preprint arXiv:1909.08053*.

Shridhar, Mohit, Lucas Manuelli, and Dieter Fox (2021). CLIPort: What and Where Pathways for Robotic Manipulation. In: *arXiv preprint arXiv: Arxiv-2109.12098*.

– (2022). Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation. In: *arXiv preprint arXiv: Arxiv-2209.05451*.

Shridhar, Mohit, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox (2020). ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, pp. 10737–10746. DOI: 10.1109/CVPR42600.2020.01075. URL: https://openaccess.thecvf.com/content%5C_CVPR%5C_2020/html/Shridhar%5C_ALFRED%5C_A%5C_Benchmark%5C_for%5C_Interpreting%5C_Grounded%5C_Instructions%5C_for%5C_Everyday%5C_Tasks%5C_CVPR%5C_2020%5C_paper.html.

Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. (2016). Mastering the Game of Go With Deep Neural Networks and Tree Search. In: *Nature* 529.7587, pp. 484–489.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. (2017). Mastering Chess and Shogi by Self-Play With a General Reinforcement Learning Algorithm. In: *arXiv preprint arXiv:1712.01815*.

Silver, Tom, Soham Dan, Kavitha Srinivas, Joshua B Tenenbaum, Leslie Pack Kaelbling, and Michael Katz (2023). Generalized Planning in PDDL Domains With Pretrained Large Language Models. In: *arXiv preprint arXiv:2305.11014*.

Singh, Ishika, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg (2023). Progprompt: Generating Situated Robot Task Plans Using Large Language Models. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 11523–11530.

Singh, Satinder, Richard L Lewis, and Andrew G Barto (2009). Where Do Rewards Come From. In: *Proceedings of the Annual Conference of the Cognitive Science Society*. Cognitive Science Society, pp. 2601–2606.

Skreta, Marta, Naruki Yoshikawa, Sebastian Arellano-Rubach, Zhi Ji, Lasse Bjørn Kristensen, Kourosh Darvish, Alán Aspuru-Guzik, Florian Shkurti, and Animesh Garg (2023). Errors Are Useful Prompts: Instruction Guided Task Programming With Verifier-Assisted Iterative Prompting. In: *arXiv preprint arXiv: Arxiv-2303.14100*.

Skrynnik, Alexey, Aleksey Staroverov, Ermek Aitygulov, Kirill Aksenov, Vasilii Davydov, and Aleksandr I. Panov (2021). Hierarchical Deep Q-Network From Imperfect Demonstrations in Minecraft. In: *Cognitive Systems Research* 65, pp. 74–78. DOI: 10.1016/j.cogsys.2020.08.012. URL: https://doi.org/10.1016/j.cogsys.2020.08.012.

Smith, Laura, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine (2019). AVID: Learning Multi-Stage Tasks via Pixel-Level Translation of Human Videos. In: *arXiv preprint arXiv: Arxiv-1912.04443*.

Srinivasan, Krishnan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn (2020). Learning to Be Safe: Deep RL With a Safety Critic. In: *arXiv preprint arXiv: Arxiv-2010.14603*.

Srivastava, Sanjana, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, C. Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei (2021). BEHAVIOR: Benchmark for Everyday Household Activities in Virtual, Interactive, and Ecological Environments. In: *Conference on Robot Learning, 8-11 November 2021, London, UK*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, pp. 477–490. URL: https://proceedings.mlr.press/v164/srivastava22a.html.

Standish, Russell K (2003). Open-Ended Artificial Evolution. In: *International Journal of Computational Intelligence and Applications* 3.02, pp. 167–175.

Stanley, Kenneth O, Joel Lehman, and Lisa Soros (2017). Open-Endedness: The Last Grand Challenge You've Never Heard Of. In: *O'Reilly Media.*

Stengel-Eskin, Elias, Andrew Hundt, Zhuohong He, Aditya Murali, Nakul Gopalan, Matthew Gombolay, and Gregory Hager (Nov. 2022). Guiding Multi-Step Rearrangement Tasks With Natural Language Instructions. In: *Proceedings of the 5th Conference on Robot Learning*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, pp. 1486–1501. URL: https://proceedings.mlr.press/v164/stengel-eskin22a.html.

Stepputtis, Simon, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor (2020). Language-Conditioned Imitation Learning for Robot Manipulation Tasks. In: *Advances in Neural Information Processing Systems* 33, pp. 13139–13150.

Sun, Chen, Fabien Baradel, Kevin Murphy, and Cordelia Schmid (2019). Learning Video Representations Using Contrastive Bidirectional Transformer. In: *arXiv preprint arXiv: Arxiv-1906.05743*.

Sun, Shao-Hua, Te-Lin Wu, and Joseph J. Lim (2020). Program Guided Agent. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=BkxUvnEYDH.

Sutton, Richard S and Andrew G Barto (2018). Reinforcement Learning: An Introduction. 2nd. Cambridge, MA, USA: MIT Press.

Szot, Andrew et al. (2021). Habitat 2.0: Training Home Assistants to Rearrange Their Habitat. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021,*

December 6-14, 2021, Virtual. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, pp. 251–266. URL: https://proceedings.neurips.cc/paper/2021/hash/021bbc7ee20b71134d53e20206bd6feb-Abstract.html.

Tan, Weihao, Xiangyang Li, Yunhao Fang, Heyuan Yao, Shi Yan, Hao Luo, Tenglong Ao, Huihui Li, Hongbin Ren, Bairen Yi, Yujia Qin, Bo An, Libin Liu, and Guang Shi (2025). Lumine: An Open Recipe for Building Generalist Agents in 3D Open Worlds. arXiv: 2511.08892 [cs.AI]. URL: https://arxiv.org/abs/2511.08892.

Tan, Weihao, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, et al. (2024). Cradle: Empowering Foundation Agents Towards General Computer Control. In: *arXiv preprint arXiv:2403.03186*.

Tassa, Yuval, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller (2018). DeepMind Control Suite. In: *arXiv preprint arXiv: Arxiv-1801.00690*.

Tay, Yi, Mostafa Dehghani, Dara Bahri, and Donald Metzler (2020). Efficient Transformers: A Survey. In: *arXiv preprint arXiv: Arxiv-2009.06732*.

Taylor, Tim, Mark Bedau, Alastair Channon, David Ackley, Wolfgang Banzhaf, Guillaume Beslon, Emily Dolson, Tom Froese, Simon Hickinbotham, Takashi Ikegami, et al. (2016). Open-Ended Evolution: Perspectives From the OEE Workshop in York. In: *Artificial Life* 22.3, pp. 408–423.

Team, DeepMind Interactive Agents et al. (2021a). Creating Multimodal Interactive Agents With Imitation and Self-Supervised Learning. In: *arXiv preprint arXiv: Arxiv-2112.03763*.

Team, Octo Model, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. (2024). Octo: An Open-Source Generalist Robot Policy. In: *arXiv preprint arXiv:2405.12213*.

Team, Open Ended Learning, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki (2021b). Open-Ended Learning Leads to Generally Capable Agents. In: *arXiv preprint arXiv: Arxiv-2107.12808*.

Tesla, Inc. (2025). Full Self-Driving (Supervised). https://www.tesla.com/fsd. Accessed: 2025-11-22.

Thananjeyan, Brijen, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg (2021). Recovery RL: Safe Reinforcement Learning With Learned

Recovery Zones. In: *IEEE Robotics and Automation Letters* 6.3, pp. 4915–4922. DOI: 10.1109/LRA.2021.3070252. URL: https://doi.org/10.1109/LRA.2021.3070252.

Thomas, Garrett, Ching-An Cheng, Ricky Loynd, Felipe Vieira Frujeri, Vibhav Vineet, Mihai Jalobeanu, and Andrey Kolobov (2023). PLEX: Making the Most of the Available Data for Robotic Manipulation Pretraining. In: *Conference on Robot Learning*. PMLR, pp. 2624–2641.

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample (2023). LLaMA: Open and Efficient Foundation Language Models. In: *arXiv preprint arXiv: Arxiv-2302.13971*.

Towers, Mark, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. (2024). Gymnasium: A Standard Interface for Reinforcement Learning Environments. In: *arXiv preprint arXiv:2407.17032*.

Toyama, Daniel, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup (2021). AndroidEnv: A Reinforcement Learning Platform for Android. In: *arXiv preprint arXiv: Arxiv-2105.13231*.

Toyer, Sam, Rohin Shah, Andrew Critch, and Stuart Russell (2020). The MAGICAL Benchmark for Robust Imitation. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. URL: https://proceedings.neurips.cc/paper/2020/hash/d464b5ac99e74462f321c06ccacc4bff-Abstract.html.

Tschannen, Michael, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. (2025). Siglip 2: Multilingual Vision-Language Encoders With Improved Semantic Understanding, Localization, and Dense Features. In: *arXiv preprint arXiv:2502.14786*.

Tsimpoukelli, Maria, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill (2021). Multimodal Few-Shot Learning With Frozen Language Models. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, Virtual*. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, pp. 200–212. URL: https://proceedings.neurips.cc/paper/2021/hash/01b7575c38dac42f3cfb7d500438b875-Abstract.html.

Vinyals, Oriol, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, et al. (2019a). Alphastar: Mastering the Real-Time Strategy Game Starcraft Ii. In: *DeepMind Blog* 2.

Vinyals, Oriol, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. (2019b). Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. In: *Nature* 575.7782, pp. 350–354.

Vølske, Michael, Martin Potthast, Shahbaz Syed, and Benno Stein (Sept. 2017). TL;DR: Mining Reddit to Learn Automatic Summarization. In: *Proceedings of the Workshop on New Frontiers in Summarization*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 59–63. DOI: 10.18653/v1/W17-4508. URL: https://aclanthology.org/W17-4508.

Volum, Ryan, Sudha Rao, Michael Xu, Gabriel DesGarennes, Chris Brockett, Benjamin Van Durme, Olivia Deng, Akanksha Malhotra, and Bill Dolan (July 2022). Craft an Iron Sword: Dynamically Generating Interactive Game Characters by Prompting Large Language Models Tuned on Code. In: *Proceedings of the 3rd Wordplay: When Language Meets Games Workshop (Wordplay 2022)*. Seattle, United States: Association for Computational Linguistics, pp. 25–43. URL: https://aclanthology.org/2022.wordplay-1.3.

Walke, Homer, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine (2023). BridgeData V2: A Dataset for Robot Learning at Scale. In: *Conference on Robot Learning (CoRL)*.

Wan Team (2025). Wan: Open and Advanced Large-Scale Video Generative Models. In.

Wang, Guanzhi, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar (2023a). Voyager: An Open-Ended Embodied Agent With Large Language Models. In: *arXiv preprint arXiv:2305.16291*.

Wang, Huaxiaoyue, Gonzalo Gonzalez-Pumariega, Yash Sharma, and Sanjiban Choudhury (2023b). Demo2Code: From Summarizing Demonstrations to Synthesizing Code via Extended Chain-of-Thought. In: *arXiv preprint arXiv:2305.16744*.

Wang, Liyuan, Xingxing Zhang, Hang Su, and Jun Zhu (2023c). A Comprehensive Survey of Continual Learning: Theory, Method and Application. In: *arXiv preprint arXiv: Arxiv-2302.00487*.

Wang, Rui, Joel Lehman, Jeff Clune, and Kenneth O. Stanley (2019). Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions. In: *arXiv preprint arXiv: Arxiv-1901.01753*.

Wang, Rui, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeff Clune, and Kenneth O. Stanley (2020). Enhanced POET: Open-Ended Reinforcement Learning Through Unbounded Invention of Learning Challenges and Their Solutions. In: *arXiv preprint arXiv: Arxiv-2003.08536*.

Wang, Su, Ceslee Montgomery, Jordi Orbay, Vighnesh Birodkar, Aleksandra Faust, Izzeddin Gur, Natasha Jaques, Austin Waters, Jason Baldridge, and Peter Anderson (2021). Less Is More: Generating Grounded Navigation Instructions From Landmarks. In: *arXiv preprint arXiv: Arxiv-2111.12872*.

Wang, Thomas, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel (2022a). What Language Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization? In: *Icml*. DOI: 10.48550/arXiv.2204.05832.

Wang, Wenhui, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, and Furu Wei (2022b). Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks. In: *arXiv preprint arXiv: Arxiv-2208.10442*.

Wang, Xin, Taein Kwon, Mahdi Rad, Bowen Pan, Ishani Chakraborty, Sean Andrist, Dan Bohus, Ashley Feniello, Bugra Tekin, Felipe Vieira Frujeri, Neel Joshi, and Marc Pollefeys (2023d). HoloAssist: An Egocentric Human Interaction Dataset for Interactive AI Assistants in the Real World. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Wang, Yufei, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan (2024a). RoboGen: Towards Unleashing Infinite Data for Automated Robot Learning via Generative Simulation. In: *International Conference on Machine Learning*.

Wang, Zihao, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang (2024b). Describe, Explain, Plan and Select: Interactive Planning With Large Language Models Enables Open-World Multi-Task Agents. arXiv: 2302.01560 [cs.AI]. URL: https://arxiv.org/abs/2302.01560.

Wang, Zihao et al. (2025). Game-Tars: Pretrained Foundation Models for Scalable Generalist Multimodal Game Agents. arXiv: 2510.23691 [cs.AI]. URL: https://arxiv.org/abs/2510.23691.

Wei, Jason, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus (2022a). Emergent Abilities of Large Language Models. In: *arXiv preprint arXiv: Arxiv-2206.07682*.

Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou (2022b). Chain of Thought Prompting Elicits Reasoning in Large Language Models. In: *arXiv preprint arXiv: Arxiv-2201.11903*.

Weihs, Luca, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi (2021). Visual Room Rearrangement. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, pp. 5922–5931. DOI: 10.1109/CVPR46437.2021.00586. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Weihs%5C_Visual%5C_Room%5C_Rearrangement%5C_CVPR%5C_2021%5C_paper.html.

Wen, Junjie, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. (2025). Tinyvla: Towards Fast, Data-Efficient Vision-Language-Action Models for Robotic Manipulation. In: *IEEE Robotics and Automation Letters*.

Wen, Kaiyue, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma (2024). Understanding Warmup-Stable-Decay Learning Rates: A River Valley Loss Landscape Perspective. In: *arXiv preprint arXiv:2410.05192*.

Wikipedia contributors (2022). Minecraft — Wikipedia, the Free Encyclopedia. [Online; accessed 9-June-2022]. URL: https://en.wikipedia.org/w/index.php?title=Minecraft&%20oldid=1092238294.

Wu, Hongtao, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong (2023a). Unleashing Large-Scale Video Generative Pre-Training for Visual Robot Manipulation. In: *arXiv preprint arXiv:2312.13139*.

Wu, Philipp, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel (2023b). GELLO: A General, Low-Cost, and Intuitive Teleoperation Framework for Robot Manipulators.

Wu, Sen, Hongyang R. Zhang, and Christopher Ré (2020). Understanding and Improving Information Transfer in Multi-Task Learning. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=SylzhkBtDB.

Wu, Yue, Shrimai Prabhumoye, So Yeon Min, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li (2023c). SPRING: GPT-4 Out-Performs RL Algorithms by Studying Papers and Reasoning. In: *arXiv Preprint arXiv: 2305.15486*. URL: https://arxiv.org/abs/2305.15486v2.

Xia, Fei, William B. Shen, Chengshu Li, Priya Kasimbeg, Micael Tchapmi, Alexander Toshev, Li Fei-Fei, Roberto Martín-Martín, and Silvio Savarese (2019). Interactive Gibson Benchmark (iGibson 0.5): A Benchmark for Interactive Navigation in Cluttered Environments. In: *arXiv preprint arXiv: Arxiv-1910.14442*.

Xiang, Jiannan, Guangyi Liu, Yi Gu, Qiyue Gao, Yuting Ning, Yuheng Zha, Zeyu Feng, Tianhua Tao, Shibo Hao, Yemin Shi, et al. (2024). Pandora: Towards General World Model With Natural Language Actions and Video States. In: *arXiv preprint arXiv:2406.09455*.

Xiao, Tete, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik (2022). Masked Visual Pre-Training for Motor Control. In: *arXiv preprint arXiv:2203.06173*.

Xie, Enze, Wenhai Wang, Zhiding Yu, Anima Anandkumar, José Manuel Álvarez, and Ping Luo (2021). SegFormer: Simple and Efficient Design for Semantic Segmentation With Transformers. In: *Neural Information Processing Systems*. URL: https://api.semanticscholar.org/CorpusID:235254713.

Xie, Saining, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy (2018). Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-Offs in Video Classification. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 305–321.

Xie, Yaqi, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh (2023). Translating Natural Language to Planning Goals With Large-Language Models. In: *arXiv preprint arXiv:2302.05128*.

Xu, Hu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer (2021). VideoCLIP: Contrastive Pre-Training for Zero-Shot Video-Text Understanding. In: *arXiv preprint arXiv: Arxiv-2109.14084*.

Xu, Mengdi, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua B. Tenenbaum, and Chuang Gan (2022). Prompting Decision Transformer for Few-Shot Policy Generalization. In: *arXiv preprint arXiv: Arxiv-2206.13499*.

Xu, Yang, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou (2020). LayoutLMv2: Multi-Modal Pre-Training for Visually-Rich Document Understanding. In: *arXiv preprint arXiv: Arxiv-2012.14740*.

Xu, Yiheng, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou (2019). LayoutLM: Pre-Training of Text and Layout for Document Image Understanding. In: *arXiv preprint arXiv: Arxiv-1912.13318*.

Yang, Jianwei, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, Yuquan Deng, Lars Liden, and Jianfeng Gao (2025a). Magma: A Foundation Model for Multimodal AI Agents. arXiv: 2502.13130.

Yang, Lujie, HJ Suh, Tong Zhao, Bernhard Paus Graesdal, Tarik Kelestemur, Jiuguang Wang, Tao Pang, and Russ Tedrake (2025b). Physics-Driven Data Generation for Contact-Rich Manipulation via Trajectory Optimization. In: *arXiv preprint arXiv:2502.20382*.

Yang, Zhuoyi, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. (2024). Cogvideox: Text-to-Video Diffusion Models With an Expert Transformer. In: *arXiv preprint arXiv:2408.06072*.

Yao, Shunyu, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao (2022). React: Synergizing Reasoning and Acting in Language Models. In: *arXiv preprint arXiv:2210.03629*.

Ye, Seonghyeon, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo (2025). Latent Action Pretraining From Videos. In: *The Thirteenth International Conference on Learning Representations*.

Yu, Tianhe, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn (2020). Gradient Surgery for Multi-Task Learning. In: *arXiv preprint arXiv: Arxiv-2001.06782*.

Yu, Tianhe, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Jodilyn Peralta, Brian Ichter, et al. (2023a). Scaling Robot Learning With Semantically Imagined Experience. In: *arXiv preprint arXiv:2302.11550*.

Yu, Wenhao, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. (2023b). Language to Rewards for Robotic Skill Synthesis. In: *arXiv preprint arXiv:2306.08647*.

Yuan, Haoqi, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu (2023). Plan4MC: Skill Reinforcement Learning and Planning for Open-World Minecraft Tasks. In: *arXiv Preprint arXiv: 2303.16563*. URL: https://arxiv.org/abs/2303.16563v1.

Yuan, Lu et al. (2021). Florence: A New Foundation Model for Computer Vision. In: *arXiv preprint arXiv: Arxiv-2111.11432*.

Yurtsever, Ekim, Jacob Lambert, Alexander Carballo, and Kazuya Takeda (2020). A Survey of Autonomous Driving: Common Practices and Emerging Technologies. In: *IEEE Access* 8, pp. 58443–58469.

Zakka, Kevin (July 2024). *Mink: Python Inverse Kinematics Based on MuJoCo*. Version 0.0.4. URL: https://github.com/kevinzakka/mink.

Zakka, Kevin, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi (2021). XIRL: Cross-Embodiment Inverse Reinforcement Learning. In: *arXiv preprint arXiv: Arxiv-2106.03911*.

Zeng, Andy, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Ayzaan Wahid, Vikas Sindhwani, and Johnny Lee (2020). Transporter Networks: Rearranging the Visual World for Robotic Manipulation. In: *arXiv preprint arXiv: Arxiv-2010.14406*.

Zeng, Andy, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence (2022). Socratic Models: Composing Zero-Shot Multimodal Reasoning With Language. In: *arXiv preprint arXiv: Arxiv-2204.00598*.

Zhang, Jiahui, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang (2025). ReWiND: Language-Guided Rewards Teach Robot Policies Without New Demonstrations. In: *arXiv preprint arXiv:2505.10911*.

Zhang, Tianhao, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and Pieter Abbeel (2018). Deep Imitation Learning for Complex Manipulation Tasks From Virtual Reality Teleoperation. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*.

Zhao, Tony Z., Vikash Kumar, Sergey Levine, and Chelsea Finn (2023). Learning Fine-Grained Bimanual Manipulation With Low-Cost Hardware. In: *Proceedings of Robotics: Science and Systems*.

Zhao, Wenshuai, Jorge Peña Queralta, and Tomi Westerlund (2020). Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, pp. 737–744.

Zhao, Zelin, Karan Samel, Binghong Chen, and Le Song (2021). ProTo: Program-Guided Transformer for Program-Guided Tasks. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, Virtual*. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, pp. 17021–17036. URL: https://proceedings.neurips.cc/paper/2021/hash/8d34201a5b85900908db6cae92723617-Abstract.html.

Zhen, Haoyu, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan (2024). 3d-Vla: 3D Vision-Language-Action Generative World Model. In: *arXiv preprint arXiv:2403.09631*.

Zheng, Qinqing, Amy Zhang, and Aditya Grover (2022). Online Decision Transformer. In: *arXiv preprint arXiv: Arxiv-2202.05607*.

Zheng, Ruijie, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang (2025). TraceVLA: Visual Trace Prompting Enhances Spatial-Temporal Awareness for Generalist Robotic Policies. In: *The Thirteenth International Conference on Learning Representations*.

Zhu, Henry, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine (2020a). The Ingredients of Real-World Robotic Reinforcement Learning. In: *arXiv preprint arXiv:2004.12570*.

Zhu, Linchao and Yi Yang (2020b). ActBERT: Learning Global-Local Video-Text Representations. In: *arXiv preprint arXiv: Arxiv-2011.07231*.

Zhu, Yuke, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín (2020c). Robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. In: *arXiv preprint arXiv: Arxiv-2009.12293*.

Ziebart, Brian D, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. (2008). Maximum Entropy Inverse Reinforcement Learning. In: *Association for the Advancement of Artificial Intelligence*. Vol. 8. Chicago, IL, USA, pp. 1433–1438.