

From Cells to Functional Tissue Units: Scale-Aware Deep Learning for Biological Microscopy Segmentation

Thesis by
Qilin Li

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



CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2026
Defended May 21, 2026

© 2026

Qilin Li

ORCID: 0000-0002-7191-8965

All rights reserved except where otherwise noted

ACKNOWLEDGEMENTS

I would first like to express my deepest gratitude to my advisor, Professor David Van Valen, for his continuous support, guidance, and trust throughout my PhD. He gave me the freedom to explore broadly and the confidence to pursue difficult questions with independence and persistence. His generous and unwavering support made it possible for me to grow not only as a scientist and engineer, but also as a person.

My sincere gratitude also goes to my thesis committee members, Professor Pietro Perona, Professor Georgia Gkioxari, and Professor Yisong Yue. Professor Perona has advised and guided me since my master's studies, introducing me to the expansive and beautiful world of computer vision and helping me recognize the possibilities of cross-disciplinary research at the intersection of medical imaging, biology, and artificial intelligence. Professor Gkioxari and Professor Yue offered encouragement, insight, and invaluable feedback that helped me sharpen my scientific questions, refine my technical direction, and think more clearly about the broader significance of my work.

I owe special thanks to Dr. Oisín Mac Aodha, whose mentorship was generous, patient, and unstinting. We worked through a wide range of problems, from the design of research projects to implementation details. His thoughtful guidance gave me a practical education in how to conduct research: how to ask precise questions, build carefully, revise patiently, and persist through uncertainty. I am also thankful to Professor Konstantin Zuev for offering me valuable teaching opportunities, through which I gained experience, confidence, and a stronger ability to communicate complex ideas to others.

My postdoctoral colleagues, Dr. Uriah Israel and Dr. Markus Marks, and my fellow students and colleagues in the lab made daily research life both productive and memorable. Together, we ran experiments, debugged code, maintained servers, discussed ideas, solved technical and theoretical problems, and shared happy hours. Those ordinary days became some of the most meaningful parts of my graduate experience.

I was also fortunate to have the companionship of friends I met at different stages of my life. Their support carried me through many parts of my PhD. Our conversations ranged widely across computers, games, photography, health and well-being, and the small joys of everyday discovery, from learning how to photograph a beautiful

moon to watching the squirrels around campus. Some of the most memorable moments came when friendship and intellectual curiosity met: exploring large-scale data processing with my friends, Professors Kaisa Cui and Qi Yang; discussing the expanding boundaries of AI applications; and sharing the unexpected delight of finding moments when large language models were powerful, useful, and sometimes wonderfully wrong. These conversations reminded me that research is not only a professional pursuit, but also a human one, shaped by curiosity, humor, and companionship.

My warmest thanks go to my parents, Mr. Shiyang Li and Mrs. Yanqin Xie, and to my family, whose unconditional love and support have been my strongest foundation. Their encouragement helped me maintain the strength, clarity, and resilience needed to overcome the most difficult moments of this journey.

Caltech gave me the opportunity to study, learn, and grow in a remarkable community. During these years, we faced not only academic challenges, but also the pandemic, wildfires, and many uncertainties. Going through these experiences together made my time at Caltech especially meaningful.

Finally, I would like to thank myself for continuing to move forward with patience, resilience, and hope. This journey was not always easy, but it taught me to return to the work with steadiness, humility, and trust. I still remember the words from Dr. Medina: "You are here to be successful, not to be lost." Those words stayed with me through difficult moments and reminded me to keep faith in the path I was building. As I complete this thesis, I carry forward not only the knowledge I gained, but also a deeper appreciation for persistence, kindness, and growth.

ABSTRACT

Biological microscopy turns cells, tissues, and molecular labels into images, but quantitative biology depends on converting those images into reliable objects. Segmentation defines the cell boundaries, nuclei, functional tissue units (FTUs), and background regions from which morphology, spatial neighborhoods, tissue organization, and disease-associated microenvironments are measured. This step remains difficult because biological objects are dense, small, heterogeneous in appearance, expensive to annotate, and often embedded in gigapixel-scale images with substantial stain and scale variation. General computer-vision models provide useful architectures and representations, but they must be adapted before they can reliably support biological image analysis.

This thesis addresses these challenges through four connected contributions. First, it establishes a cell preprocessing and evaluation workflow including microscopy annotation conversion, tiling, normalization, small-object mask fidelity, and object-level evaluation. Second, it studies Equivariant Mask2Former, showing that finite rotation/reflection-equivariant feature processing can help in limited-label or weak-augmentation settings, but carries substantial computational cost. Third, it contributes to CellSAM, a detector-guided cell segmentation framework in which CellFinder supplies automatic bounding-box prompts for SAM-based mask decoding across diverse cellular imaging modalities. Fourth, it introduces FTURader, a scale-dynamic FTU segmentation pipeline that combines H&E stain normalization, FTUFinder detection, dynamic scale control, and SAM2-based mask decoding.

Together, these contributions bridge cell-scale and FTU-scale microscopy segmentation. They show that reliable biological segmentation depends on controlled data representation, metrics matched to the biological question, prompt generators adapted to dense microscopy, and scale-aware inference when the target changes from cells to anatomy. The resulting perspective is that segmentation should be designed and evaluated as a measurement system for specific biological units, not simply as a mask-prediction task.

PUBLISHED CONTENT AND CONTRIBUTIONS

1. Markus Marks et al. “CellSAM: A Foundation Model for Cell Segmentation”. In: *Nature Methods* (2025). DOI: [10.1038/s41592-025-02879-w](https://doi.org/10.1038/s41592-025-02879-w). arXiv: [2311.11004](https://arxiv.org/abs/2311.11004).
Q.L performed algorithm design for CellFinder and CellSAM; implemented CellFinder; carried out experiments and evaluations; performed model benchmarking; developed data pipelines; performed data engineering; and participated in the writing of the manuscript.

TABLE OF CONTENTS

Acknowledgements	iii
Abstract	v
Published Content and Contributions	vi
Table of Contents	vi
List of Illustrations	ix
List of Tables	xi
Chapter I: Introduction	1
1.1 Biological Motivation: From Cells to Functional Tissue Units	1
1.2 Why Biological Microscopy Segmentation Is Difficult	2
1.3 Data and Evaluation as the Foundation for Reliable Segmentation	3
1.4 General Segmentation Models and Their Limits in Microscopy	4
1.5 Domain-Specific Microscopy Segmentation Models	5
1.6 Learning Under Limited Annotation	6
1.7 From Specialist Models to Biological Foundation Models	6
1.8 From Cell Segmentation to FTU Segmentation	7
1.9 Motivation and Organization of This Thesis	7
Chapter II: Cell Preprocessing and Evaluation Workflow for Consistent Segmentation Studies	13
2.1 Introduction	13
2.2 Results	14
2.3 Methods and Evaluation Protocol	21
2.4 Discussion	23
Chapter III: Equivariant Mask2Former for Cell Segmentation	27
3.1 Introduction	27
3.2 Related Work	29
3.3 Method	30
3.4 Experiments	36
3.5 Results	39
3.6 Discussion	43
Chapter IV: CellSAM: A Foundation Model for Cell Segmentation Across Modalities	48
4.1 Introduction	48
4.2 Results	51
4.3 Discussion	58
4.4 Methods	62
Chapter V: FTURader: Functional Tissue Unit Segmentation with Scale-Dynamic Adaptation	86
5.1 Introduction	86
5.2 Results	87

5.3 Discussion	94
5.4 Methods	95
5.5 Extended Data	105
Chapter VI: Conclusion and Future Directions	113
6.1 Thesis Synthesis	113
6.2 Contributions Across Chapters	114
6.3 Cross-Cutting Principles	115
6.4 Limitations	115
6.5 Future Directions	116
6.6 Final Perspective	117
Index	120

LIST OF ILLUSTRATIONS

<i>Number</i>	<i>Page</i>
1.1 Whole-body to single-cell hierarchy	2
2.1 Cell preprocessing and evaluation workflow	15
2.2 Small-object COCO polygon decoding error	18
2.3 Example dataset statistics stored in an exported COCO JSON file . .	20
3.1 Overview of the Equivariant Mask2Former workflow	27
3.2 Architecture of Equivariant Mask2Former	32
3.3 C. elegans evaluation curves without augmentation	39
3.4 C. elegans evaluation curves with augmentation	40
3.5 TissueNet evaluation curves without augmentation	41
3.6 C. elegans qualitative segmentation results	42
3.7 TissueNet qualitative segmentation results	43
4.1 CellSAM: a foundational model for cell segmentation	52
4.2 CellSAM is a strong generalist model for cell segmentation	54
4.3 CellSAM enables diverse bioimage analysis workflows	59
4.4 SAM prompting strategy comparison	68
4.5 CellSAM generalist compared with Cellpose cyto3	69
4.6 Generalist and specialist model comparison	70
4.7 LIVECell zero-shot and few-shot performance	71
4.8 LIVECell ten-shot qualitative results	73
4.9 Zero-shot CellSAM generalization to electron microscopy biofilms .	74
4.10 Inference times for CellSAM and Cellpose	76
4.11 CellSAM training curves	76
5.1 FTURader architecture for detector-prompted FTU segmentation . . .	88
5.2 FTUFinder localization and scale-estimation stage	90
5.3 Instance and COCO scores for FTU segmentation	92
5.4 FTUFinder bounding-box F1 across IoU thresholds	92
5.5 Per-image pixel Dice and IoU distributions	93
5.6 Pixel semantic scores compared with Team 1	93
5.7 Macenko stain normalization and H/E channel decomposition	96
5.8 FTUFinder training loss and validation F1	99
5.9 FTUFinder validation F1 without dynamic scaling	100

5.10	SAM2-based mask-decoder training loss	102
5.11	Qualitative kidney test-set visualization	106
5.12	Qualitative large-intestine test-set visualization	107
5.13	Qualitative lung test-set visualization	108
5.14	Qualitative prostate test-set visualization	109
5.15	Qualitative spleen test-set visualization	110

LIST OF TABLES

<i>Number</i>	<i>Page</i>
1.1 Thesis roadmap	8
2.1 Mask encoding changes COCO AP on a TissueNet subset	18
2.2 Parallel preprocessing performance on the 30 FTU dataset variants	21
3.1 Datasets for Equivariant Mask2Former	37
3.2 C. elegans instance segmentation results	40
3.3 TissueNet instance segmentation results	41
3.4 TissueNet training cost comparison	43
4.1 CellSAM dataset statistics	75
5.1 FTU dataset split statistics	89
5.2 FTUFinder training hyperparameters	98
5.3 SAM2-based mask-decoder training hyperparameters	101
5.4 Validation-selected FTUFinder score thresholds	103

Chapter 1

INTRODUCTION

1.1 Biological Motivation: From Cells to Functional Tissue Units

Biological microscopy converts tissues, cells, and molecular labels into images that can be inspected, quantified, and compared. Its scientific value depends on the ability to connect measurements across scale. Cells are the fundamental unit for many questions in cell biology and spatial omics: where a cell is located, which markers it expresses, what morphology it has, and which neighboring cells form its local microenvironment. Yet organ function is rarely explained by isolated cells alone. It emerges from repeated multicellular structures embedded in tissue architecture.

Functional tissue units (FTUs) occupy this intermediate scale. In human atlas efforts, FTUs are used to describe recurring tissue structures that connect local cellular composition to organ-specific function, such as glomeruli in kidney, crypts in colon, glandular acini in prostate, white pulp in spleen, and alveolar regions in lung¹⁻⁵.

Segmentation is the computational step that turns a microscopy image into measurable biological objects. Errors in the segmentation stage propagate into downstream biology: an inaccurate cell boundary can distort single-cell measurements, while a missed or merged FTU can alter object counts, size distributions, and tissue-level interpretation^{3,6}.

This thesis argues that reliable biological segmentation requires four linked forms of adaptation: controlled data representation, evaluation matched to the biological object, model design matched to the annotation regime, and scale-aware prompting when the target moves from cells to tissue structures. It follows a cell-to-FTU trajectory. It begins with the infrastructure required to compare biological segmentation models reliably, examines data-efficient model design under limited annotation, moves to foundation-model cell segmentation with broader heterogeneous data, and finally extends the segmentation problem from cell instances to larger functional tissue units.

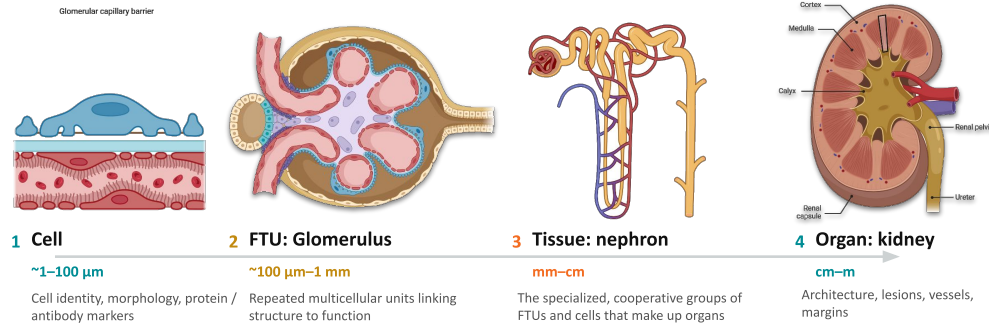


Figure 1.1: A multiscale hierarchy from whole-body macro-anatomy to the single-cell level⁷. At the highest level are organ systems, whose individual organs contain diverse tissues. Tissues, in turn, consist of repeating structures known as functional tissue units (FTUs). Finally, FTUs and tissues are composed of different cell types that provide the cellular basis for tissue organization and function.⁸.

1.2 Why Biological Microscopy Segmentation Is Difficult

Biological microscopy segmentation differs from ordinary natural-image segmentation in several linked ways. First, biological objects are often dense, small, and visually repetitive. A single field of view can contain hundreds or thousands of cells. Object identity matters: a mask that merges two cells may still cover much of the correct foreground, but it no longer supports single-cell quantification. This is why later chapters emphasize object-level matching rather than foreground overlap alone.

Second, microscopy data are heterogeneous. Images span modalities such as fluorescence, brightfield, phase contrast, histology, multiplexed protein imaging, spatial-transcriptomics imaging, and electron microscopy. Channels have biological meaning rather than generic color meaning, and contrast depends on staining, optics, sample preparation, scanner behavior, and tissue condition. A method trained on one modality can fail when the same biological object appears under a different marker or acquisition protocol⁹⁻¹¹. This motivates controlled preprocessing and domain-adapted prompting.

Third, object scale varies even when the microscope or whole-slide scanner has a fixed sampling resolution. Histology slides are commonly digitized at fixed objective magnifications such as $20\times$ or $40\times$, which set the microns-per-pixel spacing for the image acquisition¹². Within that fixed pixel grid, cells, bacteria, and FTUs occupy very different pixel ranges. For FTUs, candidate diameters in a multi-organ dataset range from approximately 30 to nearly 2000 pixels³. Whole-slide

histology compounds the problem because the image is too large to process directly at full resolution; it must be tiled, downsampled, dynamically resized, or processed hierarchically. This scale variation motivates the FTURader design in Chapter 5.

Fourth, biological annotation is expensive. For FTUs, a trained pathologist needs ca. 50 min to segment 200 FTUs, or about 15 s per FTU; for images containing a few hundred FTUs, this can correspond to roughly US\$10–US\$100 per image depending on expert labor cost and review workflow^{2,3}. Cell labels are cheaper individually but scale poorly: dense cell instance labels can cost about US\$0.01 per label, and microscopy images may contain hundreds to thousands of cells, making a fully labeled image cost on the order of US\$10¹³. This annotation burden contrasts with the scale of general foundation-model data: SAM’s SA-1B dataset contains about 11 million images and 1.1 billion masks¹⁴. Chapters 3 and 4 respond to this burden in complementary ways: one uses an explicit geometric prior, while the other uses broader data and detector-guided prompting.

These properties explain why general-purpose vision models are not automatically applicable to biological segmentation. Natural-image pretraining offers useful representations and architectures, but microscopy still requires domain-specific data handling, scale control, and color normalization.

1.3 Data and Evaluation as the Foundation for Reliable Segmentation

The first technical premise of this thesis is that model comparison is only meaningful when data conversion and evaluation are controlled. Biological segmentation datasets arrive as dense label masks, polygon annotations, run-length encoded masks, or platform-specific annotation files. Computer-vision frameworks often expect images and masks to be organized in standard formats, such as COCO format¹⁵, Pascal VOC¹⁶, YOLO detection labels¹⁷, or SA-1B mask records¹⁴. The conversion between these representations is not straightforward, because general dataset formats may not be optimized for small objects.

For a predicted object p and a ground-truth object g , common pixel-overlap scores include

$$\text{Dice}(p, g) = \frac{2|p \cap g|}{|p| + |g|}, \quad \text{IoU}(p, g) = \frac{|p \cap g|}{|p \cup g|}.$$

These metrics are useful but incomplete. Instance segmentation also requires matching predicted and ground-truth objects, deciding which objects are true positives, false positives, or false negatives, and aggregating precision, recall, F1, average precision, or average recall across thresholds and images. This distinction is especially

important for cells and FTUs because split, merged, or duplicated objects can be biologically unacceptable even when pixel overlap appears reasonable.

Chapter 2 therefore establishes the Cell Preprocessing and Evaluation Workflow for the rest of the thesis. It focuses on data conversion, small-object mask fidelity, tiling, normalization, dataset provenance, dataset merging, and harmonized evaluation metrics. Later chapters use this infrastructure-facing view as the basis for model development rather than treating preprocessing as an invisible prelude.

1.4 General Segmentation Models and Their Limits in Microscopy

Modern segmentation methods provide the technical vocabulary used throughout this thesis. U-Net established the encoder–decoder architecture with skip connections that remains central to biomedical semantic segmentation, especially in limited-data settings¹⁸. Mask R-CNN framed instance segmentation as detection followed by mask prediction¹⁹. DETR introduced end-to-end set prediction with learned object queries²⁰, and Mask2Former unified semantic, instance, and panoptic segmentation through query-based mask decoding and masked attention²¹.

Promptable foundation models changed the segmentation interface. The Segment Anything Model (SAM) predicts masks from points, boxes, or masks and can transfer to new image domains more flexibly than closed-category models¹⁴. SAM2 extends promptable segmentation to images and videos and is relevant to the mask-decoding design explored later in this thesis²².

Despite these advances, direct transfer from natural images to microscopy is limited by dense repeated instances, modality-specific contrast, ambiguous boundaries, and scale variation. Recent cell-segmentation benchmarks make this limitation explicit. A systematic evaluation of 18 methods reported that vanilla SAM underperformed for cell segmentation, likely because the pretrained foundation model was not fine-tuned on the cell training images²³. The limitation persists for newer off-the-shelf SAM-family models: a later benchmark compared general-purpose SAM, SAM2, and SAM3 with microscopy-adapted variants and found that competitive performance required cell-specific adaptation, especially automatic prompt generation²⁴. Thus, vanilla SAM-family models do not perform well as out-of-the-box cell segmenters.

The adapted methods make this contrast clearer. Cellpose-SAM adapts a pretrained SAM backbone to the Cellpose framework²⁵; CellSAM couples SAM-based mask decoding with an automated cell detector that supplies box prompts¹³; and μ SAM

adapts SAM to microscopy annotation and segmentation workflows¹¹. These examples support the thesis view that general vision models are useful architectural resources, but biological cell segmentation still requires domain-specific prompting, training, or data handling.

1.5 Domain-Specific Microscopy Segmentation Models

Before biological foundation models became practical, many successful microscopy methods encoded biological assumptions directly into the model output. Cellpose predicts flows that move pixels toward object centers, enabling separation of many touching cellular objects²⁶. StarDist represents nuclei and approximately round cells as star-convex polygons, which is powerful when object geometry matches that assumption²⁷. Mesmer uses large-scale tissue-cell annotation to segment whole cells in multiplex tissue images²⁸. Omnipose modifies flow-based segmentation for elongated and irregular bacterial morphology²⁹.

These methods show that biological priors matter. A center-flow representation, radial boundary model, nuclear/cytoplasmic channel assumption, or tissue-specific training set can produce strong practical performance. The same priors can also limit transfer. A model optimized for round nuclei may struggle with branched cells; a tissue-cell model may depend on particular markers; a bacterial segmentation method may not transfer to mammalian tissue. The field therefore moved toward two complementary directions: better inductive biases for limited annotation and larger heterogeneous training corpora for generalist models.

Domain-specific large segmentation models now bridge handcrafted biological priors and fully general promptable vision. In general medical imaging, MedSAM³⁰ and SAM-Med2D³¹ adapt SAM to large 2D medical segmentation corpora, SAM-Med3D³² targets volumetric images, and MedicoSAM³³ studies robust medical fine-tuning; BiomedParse³⁴ broadens this direction to joint segmentation, detection, and recognition across biomedical modalities. In microscopy and pathology, μ SAM¹¹ provides interactive and automatic microscopy workflows, PathoSAM³⁵ specializes SAM-related segmentation for histopathology, and CellViT/CellViT++^{36,37} represent a related transformer/foundation-model lineage for nuclei segmentation and classification. Cell-scale systems then specialize prompts or outputs: CellSAM¹³ uses CellFinder to generate cellular box prompts for a SAM mask decoder, Cellpose-SAM²⁵ brings SAM-scale representation learning into the Cellpose ecosystem, and SAMCell³⁸ fine-tunes SAM for label-free cell boundaries with a distance-map out-

put. Datasets including TissueNet, LIVECell, Omnipose, DeepBacs, BriFiSeg, Phase400, YeaZ, and YeastNet illustrate the diversity of cell segmentation benchmarks. Together, these models and datasets show that biological and medical foundation segmentation usually keeps a large visual backbone but changes the training distribution, prompting strategy, decoder target, or post-processing to match medical, histological, or cellular objects.

1.6 Learning Under Limited Annotation

Biological annotations are expensive because they require expertise. To make things worse, the professionals' total available work hours are limited. Classical responses include augmentation, transfer learning, self-configuring pipelines, and weak or semi-supervised learning. A more architectural response is to encode equivariances that are natural for microscopy.

For a segmentation function f and a chosen transformation group G , this thesis follows the group-equivariant T_g/T'_g notation used in Chapter 3:

$$f(T_g x) = T'_g f(x), \quad g \in G.$$

Here T_g denotes the operation associated with g on the input image x , and T'_g denotes the corresponding but distinct operation on the output mask or feature value $f(x)$. The prime emphasizes that the input and output transformations are linked by the same group element but act on different representations. Translation equivariance is already central to convolutional networks. Rotation and reflection equivariance can be valuable when cells, nuclei, bacteria, or tissue structures have no canonical in-plane orientation^{39,40}. In Chapter 3, G refers to the finite rotation/reflection group $C_8 \rtimes C_2$ implemented by `flipRot2dOnR2(N=8)`, while spatial translations are handled by convolution. Thus the implemented prior is finite rotation/reflection equivariance rather than equivariance to the full continuous Euclidean group $E(2)$.

Chapter 3 develops this idea through Equivariant Mask2Former. This chapter represents the small-data and inductive-bias stage: before relying fully on large heterogeneous foundation-model training, it asks whether finite rotation/reflection-equivariant segmentation can make better use of limited biological labels.

1.7 From Specialist Models to Biological Foundation Models

Promptable foundation-model segmentation promises a different route: instead of designing a separate specialist model for every imaging condition, train or adapt a promptable model across a broad biological distribution. Medical and microscopy

adaptations of SAM show the potential of this direction, while also showing that domain adaptation remains necessary^{11,30}.

CellSAM is the central example for this thesis. It adapts promptable segmentation to dense cell segmentation by pairing CellFinder, a detector that automatically proposes cellular bounding-box prompts, with a mask decoder that produces instance masks¹³.

The conceptual importance of CellSAM is that it changes the role of prompting. Manual prompting is infeasible when an image contains many cells, and uniform grid prompting may miss dense or irregularly distributed objects. Detector-guided prompting converts a general mask decoder into an automatic cell segmentation system.

1.8 From Cell Segmentation to FTU Segmentation

Cell segmentation is necessary for single-cell biology, but it is not sufficient for tissue-level interpretation. FTUs are larger, more heterogeneous, and more context-dependent than cells. Their boundaries may be defined by epithelial organization, luminal structure, basement membrane, vascular compartments, local cell density, or tissue texture. Their interiors may be hollow or weakly stained. Neighboring FTUs may be separated by subtle anatomical cues.

Chapter 5 therefore moves from cell-level promptable segmentation toward FTU-Rader, a scale-dynamic FTU pipeline. The implemented design uses whole-slide tiling and Macenko stain normalization⁴¹. FTUFinder localizes candidate units and estimates scale⁴²; dynamic crop scaling presents objects to the mask decoder at a suitable size; and the SAM2 promptable mask decoder generates the final prediction²².

1.9 Motivation and Organization of This Thesis

The chapters that follow form a progression from controlled data infrastructure to multiscale biological segmentation.

Chapter 2 addresses the data and evaluation foundation. It develops the Cell Preprocessing and Evaluation Workflow for converting biological annotations, preserving small-object masks, tiling large images, normalizing heterogeneous inputs, and reporting harmonized object-level metrics. This chapter makes later model comparisons controlled and interpretable.

Chapter 3 studies segmentation under limited annotation. It introduces Equivariant Mask2Former as a way to connect query-based segmentation with finite rota-

tion/reflection equivariance in microscopy. Its role in the thesis is to test whether rotation-aware inductive bias can improve data efficiency before relying on large-scale foundation-model training.

Chapter 4 presents CellSAM as the foundation model stage for cell segmentation. It focuses on automatic bounding box prompting and broad cell segmentation across biological imaging domains. The chapter provides the cell-scale generalist model that motivates the final transition to tissue-unit segmentation.

Chapter 5 extends the thesis from cells to FTUs. It treats functional tissue units as the biological scale that links cellular organization to organ architecture, and develops a scale-dynamic detector-prompted approach for segmenting FTUs in whole-slide tissue images. This chapter closes the thesis by moving from single-cell segmentation to tissue-unit-level biological measurement.

Table 1.1: Thesis roadmap

Chapter	Scale / problem	Role in the cells-to-FTUs narrative
Chapter 2	Data conversion and evaluation	Establishes reliable preprocessing, mask fidelity, and object-level metrics.
Chapter 3	Cell segmentation under limited labels	Tests geometric inductive bias for data-efficient biological segmentation.
Chapter 4	Generalist cell segmentation	Presents detector-guided promptable segmentation for cell instances.
Chapter 5	FTU segmentation	Extends promptable segmentation to tissue-unit objects with scale-dynamic adaptation.

References

1. HuBMAP Consortium. “The human body at cellular resolution: the NIH Human Biomolecular Atlas Program”. In: *Nature* 574.7777 (Oct. 9, 2019), pp. 187–192. DOI: 10.1038/s41586-019-1629-x.
2. Yashvardhan Jain et al. “Segmentation of human functional tissue units in support of a Human Reference Atlas”. In: *Communications Biology* 6.1 (July 19, 2023), p. 717. DOI: 10.1038/s42003-023-04848-5.
3. Yashvardhan Jain et al. “Segmenting functional tissue units across human organs using community-driven development of generalizable machine learning algorithms”. In: *Nature Communications* 14.1 (Aug. 3, 2023), p. 4656. DOI: 10.1038/s41467-023-40291-0.

4. Katy Börner et al. “Construction, deployment, and usage of the Human Reference Atlas”. In: *Nature Methods* 22 (2025), pp. 409–418. doi: 10.1038/s41592-024-02563-5.
5. Supriya Bidanta et al. “Functional tissue units in the Human Reference Atlas”. In: *Nature Communications* 16.1 (Feb. 11, 2025), p. 1526. doi: 10.1038/s41467-024-54591-6.
6. Viktor Petukhov et al. “Cell segmentation in imaging-based spatial transcriptomics”. In: *Nature Biotechnology* 40.3 (Mar. 2022), pp. 345–354. doi: 10.1038/s41587-021-01044-w.
7. Christian Kurts et al. “Kidney immunology from pathophysiology to clinical translation”. In: *Nature Reviews Immunology* 25 (2025), pp. 460–476. doi: 10.1038/s41577-025-01131-y.
8. BioRender. *BioRender*. Figure created with BioRender.com. 2026. URL: <https://www.biorender.com/>.
9. Christoffer Edlund et al. “LIVECell—A large-scale dataset for label-free live cell segmentation”. In: *Nature Methods* 18.9 (Sept. 2021), pp. 1038–1045. doi: 10.1038/s41592-021-01249-6.
10. Jun Ma et al. “The multimodality cell segmentation challenge: toward universal solutions”. In: *Nature Methods* 21.6 (June 2024), pp. 1103–1113. doi: 10.1038/s41592-024-02233-6.
11. Anwai Archit et al. “Segment Anything for Microscopy”. In: *Nature Methods* 22.3 (Mar. 2025), pp. 579–591. doi: 10.1038/s41592-024-02580-4.
12. Joshua D. Webster and Robert W. Dunstan. “Whole-slide imaging and automated image analysis: considerations and opportunities in the practice of pathology”. In: *Veterinary Pathology* 51.1 (2014), pp. 211–223. doi: 10.1177/0300985813503570.
13. Markus Marks et al. “CellSAM: a foundation model for cell segmentation”. In: *Nature Methods* 22.12 (Dec. 8, 2025), pp. 2585–2593. doi: 10.1038/s41592-025-02879-w.
14. Alexander Kirillov et al. “Segment Anything”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4015–4026. arXiv: 2304.02643.
15. Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Springer, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48. arXiv: 1405.0312.
16. Mark Everingham et al. “The PASCAL Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338. doi: 10.1007/s11263-009-0275-4.

17. Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788. doi: 10.1109/cvpr.2016.91.
18. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer, 2015, pp. 234–241. doi: 10.1007/978-3-319-24574-4_28. arXiv: 1505.04597.
19. Kaiming He et al. “Mask R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2961–2969. doi: 10.1109/iccv.2017.322. arXiv: 1703.06870.
20. Nicolas Carion et al. “End-to-End Object Detection with Transformers”. In: *Computer Vision – ECCV 2020*. Springer, 2020, pp. 213–229. doi: 10.1007/978-3-030-58452-8_13. arXiv: 2005.12872.
21. Bowen Cheng et al. “Masked-Attention Mask Transformer for Universal Image Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1290–1299. doi: 10.1109/cvpr52688.2022.00135. arXiv: 2112.01527.
22. Nikhila Ravi et al. “SAM 2: Segment Anything in Images and Videos”. In: *International Conference on Learning Representations*. 2025. arXiv: 2408.00714.
23. Yuxing Wang et al. “A systematic evaluation of computational methods for cell segmentation”. In: *Briefings in Bioinformatics* 25.5 (2024), bbae407. doi: 10.1093/bib/bbae407.
24. Anwai Archit and Constantin Pape. “Revisiting foundation models for cell instance segmentation”. In: *Proceedings of The 9th International Conference on Medical Imaging with Deep Learning*. Vol. 315. Proceedings of Machine Learning Research. PMLR, 2026, pp. 3464–3495. arXiv: 2603.17845.
25. Marius Pachitariu, Michael Rariden, and Carsen Stringer. “Cellpose-SAM: superhuman generalization for cellular segmentation”. In: *bioRxiv* (May 1, 2025). doi: 10.1101/2025.04.28.651001.
26. Carsen Stringer et al. “Cellpose: a generalist algorithm for cellular segmentation”. In: *Nature Methods* 18.1 (Jan. 2021), pp. 100–106. doi: 10.1038/s41592-020-01018-x.
27. Uwe Schmidt et al. “Cell Detection with Star-Convex Polygons”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2018*. Springer, 2018, pp. 265–273. doi: 10.1007/978-3-030-00934-2_30. arXiv: 1806.03535.

28. Noah F. Greenwald et al. “Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning”. In: *Nature Biotechnology* 40.4 (Apr. 2022), pp. 555–565. doi: 10.1038/s41587-021-01094-0.
29. Kevin J. Cutler et al. “Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation”. In: *Nature Methods* 19.11 (Nov. 2022), pp. 1438–1448. doi: 10.1038/s41592-022-01639-4.
30. Jun Ma et al. “Segment anything in medical images”. In: *Nature Communications* 15.1 (Jan. 22, 2024). doi: 10.1038/s41467-024-44824-z.
31. Junlong Cheng et al. “SAM-Med2D”. In: *arXiv preprint arXiv:2308.16184* (2023). arXiv: 2308.16184.
32. Haoyu Wang et al. “SAM-Med3D: Towards General-purpose Segmentation Models for Volumetric Medical Images”. In: *arXiv preprint arXiv:2310.15161* (2023). arXiv: 2310.15161.
33. Anwai Archit, Luca Freckmann, and Constantin Pape. “MedicoSAM: Robust Improvement of SAM for Medical Imaging”. In: *IEEE Transactions on Medical Imaging* 45.5 (2026), pp. 1974–1985. doi: 10.1109/tmi.2025.3644811.
34. Theodore Zhao et al. “A foundation model for joint segmentation, detection and recognition of biomedical objects across nine modalities”. In: *Nature Methods* 22.1 (2025), pp. 166–176. doi: 10.1038/s41592-024-02499-w.
35. Titus Griebel, Anwai Archit, and Constantin Pape. “Segment Anything for Histopathology”. In: *arXiv preprint arXiv:2502.00408* (2025). arXiv: 2502.00408.
36. Fabian Hörst et al. “CellViT: Vision Transformers for precise cell segmentation and classification”. In: *Medical Image Analysis* 94 (May 2024), p. 103143. doi: 10.1016/j.media.2024.103143.
37. Fabian Hörst et al. “CellViT++: Energy-efficient and adaptive cell segmentation and classification using foundation models”. In: *Computer Methods and Programs in Biomedicine* 277 (2026), p. 109206. doi: 10.1016/j.cmpb.2025.109206.
38. Alexandra Dunnum VandeLoo et al. “SAMCell: Generalized label-free biological cell segmentation with segment anything”. In: *PLOS ONE* 20.9 (2025), e0319532. doi: 10.1371/journal.pone.0319532.
39. Taco Cohen and Max Welling. “Group Equivariant Convolutional Networks”. In: *Proceedings of the 33rd International Conference on Machine Learning*. PMLR, 2016, pp. 2990–2999. arXiv: 1602.07576.
40. Benjamin Chidester et al. “Rotation Equivariant and Invariant Neural Networks for Microscopy Image Analysis”. In: *Bioinformatics* 35.14 (2019), pp. i530–i537. doi: 10.1093/bioinformatics/btz353.

41. Marc Macenko et al. “A method for normalizing histology slides for quantitative analysis”. In: *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. 2009, pp. 1107–1110. doi: 10.1109/isbi.2009.5193250.
42. Isaac Robinson et al. “RF-DETR: Neural Architecture Search for Real-Time Detection Transformers”. In: *International Conference on Learning Representations*. 2026. arXiv: 2511.09554.

Chapter 2

CELL PREPROCESSING AND EVALUATION WORKFLOW FOR CONSISTENT SEGMENTATION STUDIES

2.1 Introduction

Robust cell segmentation depends on more than the accuracy of a segmentation model. It also depends on whether images, annotations, and metrics define the same measurement problem before training or evaluation begins. Microscopy datasets differ in file format, object scale, channel layout, stain distribution, annotation convention, and object density. The same field of view may be represented as a dense label image, a collection of polygons, a run-length encoded mask, or a standard annotation format such as COCO JSON¹, Pascal VOC², YOLO labels³, or SA-1B mask records⁴. If these representations are converted inconsistently, a benchmark can measure annotation artifacts rather than model behavior.

This problem is amplified in cellular images. Cells, nuclei, and bacteria can occupy only a small number of pixels, while dense tissue images may contain hundreds or thousands of annotated objects in a single field of view. General-purpose annotation formats and rasterization utilities were developed mainly for natural-image objects that often span tens to hundreds of pixels in each dimension. When these encoders are used to transcode tiny biological annotations, lossy polygon approximation or coordinate rounding during conversion can introduce substantial boundary errors. A one-pixel shift may matter little for a large natural object but can dominate the measured boundary of a small cell mask.

Metric terminology also differs between biological segmentation software and general computer-vision benchmarks. A metric that appears reasonable for semantic segmentation can hide merged cells, split cells, or duplicated detections at tile boundaries. For example, in the Cellpose family, the metric named average precision (AP) is computed from matched objects as $TP/(TP + FP + FN)$ at specified IoU thresholds^{5–8}. This differs from the conventional object-detection definition of AP as the area under a precision–recall curve, as used in COCO evaluation¹. As a result, preprocessing and evaluation are not peripheral engineering details; they define the measurement instrument used by the rest of the thesis.

The central claim of this chapter is that preprocessing and evaluation define the

measurement instrument for cell-segmentation studies. A useful workflow must preserve small-object masks, record data-transform choices, and report object-level metrics with unambiguous definitions. The Cell Preprocessing and Evaluation Workflow organizes these steps for the later chapters of this thesis. It converts heterogeneous microscopy annotations into a canonical instance representation and checks common failure modes in mask conversion. It then preprocesses large images through configurable tiling and normalization, merges datasets when needed, and reports dataset statistics. Models can also be evaluated through normalized object-level metrics derived from shared matched-object counts.

Recent cell-segmentation resources show why such a controlled workflow is needed. LIVECell⁹, TissueNet/Mesmer¹⁰, PanNuke¹¹, and the 2018 Data Science Bowl¹² each made important progress in cellular or nuclear segmentation, but they differ in modality, annotation target, file organization, and evaluation practice. The Cell Tracking Challenge further illustrates that object identity and matching rules can be as important as pixel overlap when segmentation outputs are used for downstream biological quantification¹³. A shared preprocessing and evaluation workflow makes these differences inspectable before they become hidden confounders.

This chapter does not claim that a single data format or metric is optimal for every task. Instead, it argues that each conversion, preprocessing, and evaluation choice should be explicit, checked where possible, and held constant when models are compared. Chapter 3 uses these definitions to evaluate data-efficient equivariant segmentation models. Chapter 4 relies on the same conversion and object-matching logic to compare a generalist cell segmentation model across modalities. Chapter 5 extends the same principles to whole-slide tiling, stain normalization, and instance-aware functional tissue unit segmentation.

2.2 Results

The workflow defines a shared measurement pipeline

Figure 2.1 summarizes the preprocessing and evaluation workflow used in the thesis. The workflow has a bidirectional format-conversion layer and two main modules. The conversion layer imports biological microscopy images and instance annotations, exports COCO-compatible detection/segmentation records for model training, and converts COCO annotations or predictions back into biological microscopy formats such as dense instance masks. The preprocessing module tiles and normalizes large images while preserving provenance. The evaluation module reports metric

summaries that distinguish object matching, pixel overlap, and dataset-level aggregation.

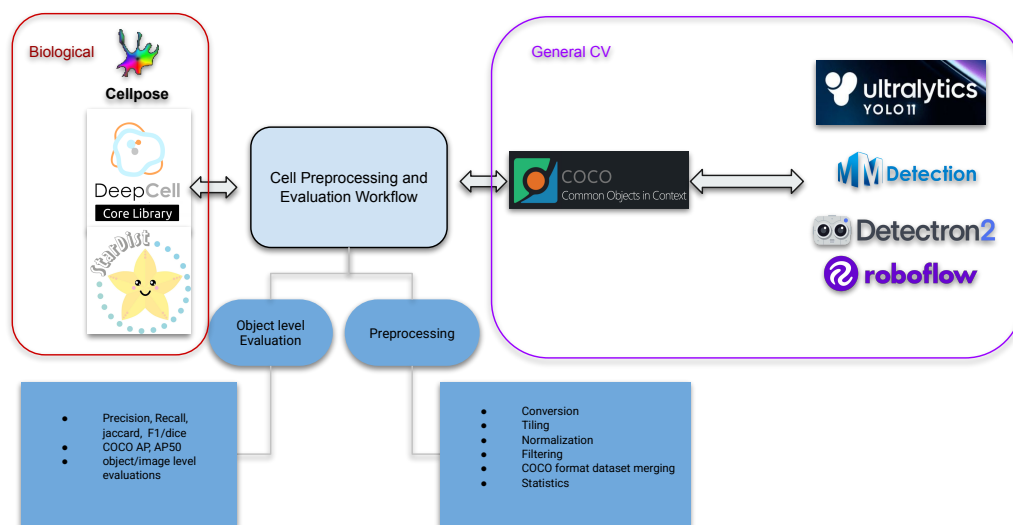


Figure 2.1: **Overview of the preprocessing and evaluation workflow.** The workflow connects biological microscopy formats and COCO-format records through a bidirectional conversion layer, then applies two main modules: preprocessing for tiling, normalization, filtering, and dataset merging, and evaluation for statistics and metrics such as precision, recall, object-level F1/Jaccard, and COCO AP/AP50 when confidence scores are available.

The design follows four principles. First, source data should be preserved as faithfully as possible while derived training formats are made consistent and traceable. Bioimage formats such as OME-TIFF and OME-NGFF address metadata and multiscale image storage, whereas machine-learning frameworks often require arrays, tiles, and object annotations in task-specific layouts^{14–16}. A preprocessing workflow therefore sits between source microscopy image formats and model-specific dataloaders.

Second, annotation conversion should be loss-aware. COCO provides a widely used representation for object detection and instance segmentation and defines standard AP/AR evaluation protocols¹. However, COCO polygons, compressed RLE, and dense label masks are not interchangeable for small biological objects. The workflow therefore treats conversion as a checkable operation rather than a neutral file-format change.

Third, preprocessing should separate biological choices from computational choices. Cropping a whole-slide image into tiles, clipping intensities, applying contrast-

limited adaptive histogram equalization (CLAHE), matching histology stains, or filtering empty tiles can be necessary for computation, but these operations alter the data presented to the model. They should therefore be parameterized, recorded, and applied consistently across experiments^{17–20}.

Fourth, evaluation should report what is biologically meaningful. Pixel overlap, object detection, instance separation, and image-level summaries answer different questions. Recent work has emphasized that segmentation metrics in bioimage analysis are frequently misinterpreted when the level of analysis is not specified²¹. The workflow therefore treats metric definitions as first-class components rather than post hoc scripts.

The contribution is a practical integration rather than a replacement for existing standards. COCO, OME, Detectron2, Cellpose, DeepCell, and StarDist each solve important parts of the segmentation workflow^{1,5,14,15,22–24}. The Cell Preprocessing and Evaluation Workflow connects these tools through a bidirectional conversion layer: biological microscopy images and masks can be exported to COCO-compatible training records, and COCO annotations or predictions can be converted back into microscopy-oriented instance masks. Around this conversion layer, the workflow provides two shared modules: preprocessing for tiling, normalization, filtering, and dataset merging, and evaluation for object-level and pixel-level metrics computed from matched-object counts.

Loss-aware conversion preserves small biological masks

The workflow uses a canonical internal representation consisting of an image array, spatial metadata when available and a set of instance records. Each instance record contains a category label, a bounding box, an area, an optional polygon representation, and either an RLE representation or a dense binary mask when exact mask fidelity is required. This representation is intentionally redundant: redundancy allows conversion outputs to be checked against each other before a dataset is exported.

Dense label images are often the most reliable source for biological instance masks because every object identity is represented at pixel resolution. They are common in biomedical segmentation datasets and can be stored as integer-valued TIFF, PNG, HDF5, or Zarr arrays. Polygon annotations are compact and convenient for manual annotation tools, and they are used by dataset formats, software frameworks, and annotation platforms such as COCO¹, Detectron2²², and the Roboflow Supervision

library²⁵. RLE stores masks losslessly relative to a raster grid and is also used in large mask datasets such as SA-1B⁴, making it useful when bit-exact object masks are required. The workflow treats dense labels or RLE as the preferred truth source when small-object boundary fidelity matters.

For compatibility with common computer-vision training and evaluation frameworks, the workflow exports datasets to COCO JSON with image records, annotation records, category metadata, and segmentation fields. Bounding boxes are derived from binary masks using a single convention for inclusive and exclusive pixel ranges. Areas are computed from the raw masks used for training or evaluation, not from a separate polygon formula, unless a specific experiment intentionally studies polygon-derived area.

Importing COCO annotations is handled symmetrically. Polygon and RLE annotations are decoded into binary masks; category IDs are mapped to biological labels; invalid or empty masks are filtered; and every annotation is assigned a unique instance identifier. The conversion step records summary statistics such as object count, mask area distribution, empty annotation count, and objects per image. These summaries are used to catch common failures before training begins.

The need for loss-aware conversion is visible in a small-object failure case. A polygon boundary shifted by one pixel can substantially change IoU, Dice score, and measured area when an object spans only a few pixels. Figure 2.2 shows this failure mode for the tested COCO polygon-decoding path. In this example, the source of the error is the COCO API `rleFrPoly()` conversion, where coordinate rounding with $+0.5$ is not bit-perfect for these small masks. In the current validation record, 24 of 100 annotations produced incorrect decoded masks. For one 15×23 pixel object shown at $8 \times$ magnification, the erroneous mask is shifted approximately one pixel toward the upper left along its boundary, corresponding to about a 20% annotation error for that object. An error of this magnitude is not a minor formatting artifact; in a small-object benchmark, it can be large enough to change the apparent quality of a model.

To estimate the possible effect of this conversion error on evaluation, the same trained model was benchmarked on the same TissueNet subset¹⁰ encoded with two mask-representation methods: one preserving bit-perfect masks with RLE and one using the problematic polygon-derived masks. The overall COCO AP decreased from 68.727 to 48.598 when the problematic polygon masks were used, and AP75 decreased by 30.794 points (Table 2.1). The larger drop at stricter IoU thresholds

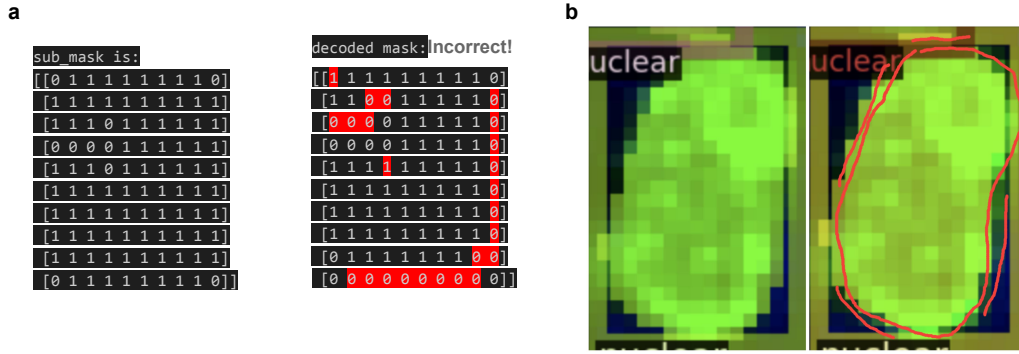


Figure 2.2: **Small-object mask error during COCO polygon decoding.** **a**, The COCO API conversion is not bit-perfect for small masks in this validation example, producing incorrect masks for 24 of 100 annotations. **b**, A 15×23 pixel object shown at $8 \times$ magnification. The left mask is the correct raster annotation, whereas the right mask is the incorrectly decoded annotation. The erroneous mask boundary is shifted approximately one pixel toward the upper left, producing about a 20% annotation error for this object.

is consistent with the visual error in Figure 2.2: small boundary shifts may have limited effect at loose matching thresholds but can strongly change whether a small object is counted as a correct instance at higher thresholds.

Table 2.1: **Mask encoding changes small-object COCO AP on a TissueNet subset.** The same trained model was evaluated on the same TissueNet subset encoded in two ways: a bit-perfect RLE mask representation and a polygon-derived mask representation affected by the decoding error shown in Figure 2.2. This comparison isolates the effect of annotation encoding: lower scores in the polygon-derived condition reflect evaluation against shifted or distorted masks rather than a different model or dataset.

Mask representation	AP	AP50	AP75	APs	APm	API
RLE bit-perfect mask	68.727	95.134	80.272	68.724	76.004	N/A
Polygon problematic mask	48.598	91.069	49.478	48.585	69.597	N/A
Difference: RLE – Polygon	+20.129	+4.065	+30.794	+20.139	+6.407	N/A

A practical workaround is to use a single rasterization convention for both directions: encode masks with `cv2.findContours()` and decode the resulting contours with `cv2.fillPoly()`. This contour-based round trip avoids the non-bit-perfect `rleFrPoly()` rounding in the tested examples. However, it is not a drop-in replacement for RLE or dense masks. In the current implementation it is safest for simple, hole-free objects; masks with holes, strong concavities, touching instances, or self-intersections should still be preserved as dense labels or RLE.

Preprocessing tiles large images into model-ready training crops

Many microscopy images are too large to process as a single tensor. Whole-slide images can be gigapixel-scale, and high-resolution fields of view can exceed GPU memory even when the biological objects are small. The preprocessing module therefore performs a one-way data-preparation step: it converts large microscopy images and their instance annotations into fixed-size crops for model training or inference. The workflow supports configurable tile size, stride, and overlap.

The tiling step also controls object density in each crop. Dense cell images can contain more annotated instances than a model can handle in one input image, for example because the detector has a limited number of queries or the mask model has a practical object limit. The workflow can therefore enforce a maximum annotation count per tile, ensuring that exported training crops do not exceed the model's capacity.

Microscopy modalities differ strongly in intensity distribution and channel semantics. Fluorescence images may contain one or more channels with sparse signal and uneven illumination. Phase-contrast and bright-field images encode cell boundaries through low-contrast interference patterns. Hematoxylin and eosin (H&E) slides vary with staining protocol, scanner, and tissue preparation. A preprocessing workflow must normalize these inputs without erasing biologically meaningful signal.

The workflow supports simple per-channel scaling, percentile clipping, histogram equalization, CLAHE, and optional stain normalization for histology images^{17–20}. These operations are not treated as automatically beneficial. Instead, the workflow provides them as selectable normalization options, allowing each experiment to choose the transformation most appropriate for its modality and modeling goal.

Dataset merging is necessary when training or comparing generalist segmentation models. The workflow merges COCO-format datasets by remapping image IDs, annotation IDs, and category IDs; copying or linking image assets; and validating that file paths, dimensions, and annotation counts remain consistent. It can filter empty tiles, invalid annotations, zero-area masks, disconnected labels, and objects outside the image boundary.

In addition to converting annotations, the workflow writes dataset-level summaries into the exported COCO JSON so that each training or evaluation split carries its own summary record. The standard COCO fields, including `info`, `licenses`,

images, categories, and annotations, are preserved. The workflow also adds summary fields such as `dataset_stats` and `image_stats`. These fields record the total number of images and annotations, the maximum number of annotations per image, the image containing that maximum count, image-intensity statistics such as mean, standard deviation, minimum, and maximum value, and the image data type. Figure 2.3 shows an example JSON tree generated by this process. Storing these statistics directly in the COCO file makes later debugging easier because preprocessing choices, dataset size and object-density summaries travel with the annotations used by downstream training and evaluation code.

```

root
├── info
│   ├── description "public_train/kidney/image"
│   ├── url ""
│   ├── version "1.0"
│   ├── year 2026
│   ├── contributor "Please refer to the paper"
│   └── date_created "2026-01-03"
├── dataset_stats
│   ├── total_annotations 337
│   ├── max_anno_per_image 7
│   └── max_anno_image_id 100000027
├── image_stats
│   ├── total_images 99
│   ├── mean [] 3 items
│   ├── std [] 3 items
│   ├── max [] 3 items
│   ├── min [] 3 items
│   └── dtype "uint8"
├── licenses [] 1 item
├── images [] 99 items
├── categories [] 1 item
└── annotations [] 337 items

```

Figure 2.3: **Example dataset statistics stored in an exported COCO JSON file.** In addition to standard COCO fields, the workflow records dataset-level annotation counts and image-level summary statistics that can be checked before model training or evaluation.

This machinery is central to the thesis arc. Chapter 4 depends on merged heterogeneous cell datasets to train and evaluate a generalist model. Chapter 5 reuses the same concepts to construct a mixed multi-organ FTU dataset for FTUFinder training and evaluation.

Parallel execution reduces preprocessing turnaround time

Large microscopy datasets make preprocessing throughput a practical bottleneck. FTUFinder, for example, requires converting many whole-slide-derived crops across a mixed multi-organ FTU dataset before model training and evaluation. These steps include image loading, mask conversion, filtering, statistics extraction, and COCO

JSON writing, all of which become expensive when repeated over thousands of tiles or annotations. The workflow supports parallel preprocessing so that independent images or crops can be processed by multiple worker processes.

Table 2.2 reports the processing time for 30 FTU dataset variants under different numbers of worker processes. These variants are the five organ-associated FTU categories used in Chapter 5 and summarized in Table 5.1, each in three official splits and two size versions (original resolution and crops of 1024×1024 pixels), giving $5 \times 3 \times 2 = 30$ preprocessing units; this chapter reports only the preprocessing runtime experiment. The benchmark was run on a server with $2 \times$ AMD EPYC 7763 64-Core processors, 2 TB DDR4 2933 MT/s memory, NVMe SSD storage, Ubuntu 24.04 and Python 3.10. Increasing from one to eight processes reduces runtime from 864.85 s to 146.55 s, a $5.90\times$ speedup with 73.8% parallel efficiency. Additional workers continue to reduce wall-clock time, reaching 93.44 s with 32 processes, although efficiency decreases as coordination and I/O overhead become more important.

Number of processes	Time (s)	Speedup vs. 1 process	Parallel efficiency
1	864.85	1.00×	100.0%
8	146.55	5.90×	73.8%
16	106.60	8.11×	50.7%
32	93.44	9.26×	28.9%

Table 2.2: **Parallel preprocessing performance on the 30 FTU dataset variants.** Increasing the number of worker processes substantially reduces wall-clock processing time, while the declining parallel efficiency reflects increasing I/O and coordination overhead at higher process counts. Dataset details are reported with the Chapter 5 FTU experiments.

2.3 Methods and Evaluation Protocol

Instance matching

Object-level evaluation begins by matching predicted masks to ground-truth masks. Let $G = \{g_i\}_{i=1}^N$ be the ground-truth objects in an image and $P = \{p_j\}_{j=1}^M$ be the predicted objects after any package-specific filtering, thresholding, or ranking. For each pair, the intersection over union (IoU) is

$$\text{IoU}(g_i, p_j) = \frac{|g_i \cap p_j|}{|g_i \cup p_j|}. \quad (2.1)$$

At a threshold τ , a prediction is a true positive (TP) if it is matched to one previously unmatched ground-truth object with $\text{IoU} \geq \tau$. Unmatched predictions are false

positives (FP), and unmatched ground-truth objects are false negatives (FN). The one-to-one matching rule is essential: it penalizes both over-segmentation, where one cell is split into several predictions, and under-segmentation, where several cells are merged into one prediction. Because different evaluation packages can apply different matching criteria, score filters, area filters, and IoU thresholds, the workflow treats TP , FP and FN as the common interface between evaluators.

Package-specific evaluators can differ in their matching and filtering rules. For COCO, DeepCell, Cellpose, and StarDist^{1,5,23,24}, the workflow therefore records the matched-object counts produced under the relevant settings and recomputes summary metrics from those counts when direct comparison is needed.

Object-level precision, recall, F1 and Jaccard

For a chosen IoU threshold and score threshold, object-level metrics are computed from the numbers of true-positive, false-positive, and false-negative instances. The workflow reports precision and recall as

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}. \quad (2.2)$$

It also reports object-level F1 and Jaccard summaries:

$$\text{F1}_{\text{object}} = \frac{2TP}{2TP + FP + FN}, \quad \text{Jaccard}_{\text{object}} = \frac{TP}{TP + FP + FN}. \quad (2.3)$$

These definitions use object counts after one-to-one instance matching, not pixel areas. Thus, the object-level F1 value reported here should not be confused with pixel-level Dice for binary mask overlap.

Pixel-level Dice and IoU are computed from foreground pixels as

$$\text{Dice}_{\text{pixel}} = \frac{2|P_{\text{fg}} \cap G_{\text{fg}}|}{|P_{\text{fg}}| + |G_{\text{fg}}|}, \quad \text{IoU}_{\text{pixel}} = \frac{|P_{\text{fg}} \cap G_{\text{fg}}|}{|P_{\text{fg}} \cup G_{\text{fg}}|}, \quad (2.4)$$

where P_{fg} is the predicted foreground pixel set and G_{fg} is the ground-truth foreground pixel set. Reporting both object-level and pixel-level metrics prevents semantic foreground overlap from being mistaken for correct instance separation.

Average precision, average recall and evaluation level

Average precision summarizes the precision–recall curve obtained by ranking predictions by confidence. In COCO instance segmentation, AP is computed across multiple IoU thresholds, commonly $\tau \in \{0.50, 0.55, \dots, 0.95\}$, and may also be

reported at specific thresholds such as AP50 and AP75¹. Average recall summarizes the maximum recall achieved under related matching and detection limits. These metrics are useful because they account for both localization accuracy and confidence ranking.

If a model produces confidence scores, this workflow reports COCO AP and AP50. Dense cellular datasets may contain more objects per image than typical natural-image benchmarks, so the maximum-detection parameter must be chosen deliberately rather than inherited blindly.

The main reporting distinction is whether a score is interpreted at the object level or the image level. Object-level evaluation asks whether individual cells are detected and matched correctly. Image-level evaluation asks how well each image performs as an experimental unit. The workflow exposes both views so that later chapters can report metrics at the level most relevant to the biological question.

2.4 Discussion

The main contribution of this chapter is to make preprocessing and evaluation choices explicit and consistently applied across the thesis. In cell segmentation, hidden data conversion steps can alter the object being measured. A model may appear to improve because empty tiles were filtered differently, because small masks were lost during polygon conversion, because tile-boundary duplicates were counted differently, or because object-level and pixel-level metrics were mixed. Standardizing these steps makes model comparisons more interpretable because the data representation and metric definitions remain fixed while the model changes.

The workflow also separates shared preprocessing and evaluation choices from chapter-specific modeling. Chapter 3 can focus on whether equivariant feature hierarchies improve low-label cell segmentation because the training data and AP definitions are controlled. Chapter 4 can compare generalist and specialist cell segmentation models across modalities because annotation conversion and object matching are held fixed. Chapter 5 can extend the same ideas to FTU segmentation, where tiling, stain normalization, instance conversion and detector-prompted mask evaluation become even more important. In this sense, the chapter supplies the measurement discipline that makes later architectural claims interpretable.

Several limitations remain. The precise behavior of COCO polygon decoding and area calculation depends on software versions, coordinate conventions and rasterization choices; these should be documented with tests. Normalization can

improve robustness but can also distort biologically meaningful intensity differences, so it should be treated as a controlled preprocessing condition rather than a default truth. The contour-based workaround for small masks may not preserve complex topology, so RLE or dense labels remain safer for masks with holes, strong concavity, or touching instances. Finally, when handling very large whole-slide images, dense binary masks can consume substantial memory; the implementation should therefore be updated to process masks block by block rather than loading full-resolution masks at once.

References

1. Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Springer, 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48. arXiv: 1405.0312.
2. Mark Everingham et al. “The PASCAL Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338. DOI: 10.1007/s11263-009-0275-4.
3. Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788. DOI: 10.1109/cvpr.2016.91.
4. Alexander Kirillov et al. “Segment Anything”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4015–4026. arXiv: 2304.02643.
5. Carsen Stringer et al. “Cellpose: a generalist algorithm for cellular segmentation”. In: *Nature Methods* 18.1 (Jan. 2021), pp. 100–106. DOI: 10.1038/s41592-020-01018-x.
6. Marius Pachitariu and Carsen Stringer. “Cellpose 2.0: how to train your own model”. In: *Nature Methods* 19.12 (Dec. 2022), pp. 1634–1641. DOI: 10.1038/s41592-022-01663-4.
7. Carsen Stringer and Marius Pachitariu. “Cellpose3: one-click image restoration for improved cellular segmentation”. In: *Nature Methods* 22.3 (Mar. 2025), pp. 592–599. DOI: 10.1038/s41592-025-02595-5.
8. Marius Pachitariu, Michael Rariden, and Carsen Stringer. “Cellpose-SAM: superhuman generalization for cellular segmentation”. In: *bioRxiv* (May 1, 2025). DOI: 10.1101/2025.04.28.651001.
9. Christoffer Edlund et al. “LIVECell—A large-scale dataset for label-free live cell segmentation”. In: *Nature Methods* 18.9 (Sept. 2021), pp. 1038–1045. DOI: 10.1038/s41592-021-01249-6.

10. Noah F. Greenwald et al. “Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning”. In: *Nature Biotechnology* 40.4 (Apr. 2022), pp. 555–565. doi: 10.1038/s41587-021-01094-0.
11. Jevgenij Gamper et al. “PanNuke Dataset Extension, Insights and Baselines”. In: *arXiv preprint arXiv:2003.10778* (2020). arXiv: 2003.10778.
12. Juan C. Caicedo et al. “Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl”. In: *Nature Methods* 16.12 (Dec. 2019), pp. 1247–1253. doi: 10.1038/s41592-019-0612-7.
13. Vladimír Ulman et al. “An objective comparison of cell-tracking algorithms”. In: *Nature Methods* 14 (2017), pp. 1141–1152. doi: 10.1038/nmeth.4473.
14. Ilya G. Goldberg et al. “The Open Microscopy Environment (OME) Data Model and XML file: open tools for informatics and quantitative analysis in biological imaging”. In: *Genome Biology* 6.5 (2005), R47. doi: 10.1186/gb-2005-6-5-r47.
15. Melissa Linkert et al. “Metadata matters: access to image data in the real world”. In: *Journal of Cell Biology* 189.5 (2010), pp. 777–782. doi: 10.1083/jcb.201004104.
16. Josh Moore et al. “OME-NGFF: a next-generation file format for expanding bioimaging data-access strategies”. In: *Nature Methods* 18 (2021), pp. 1496–1498. doi: 10.1038/s41592-021-01326-w.
17. Erik Reinhard et al. “Color transfer between images”. In: *IEEE Computer Graphics and Applications* 21.5 (Sept. 2001), pp. 34–41. doi: 10.1109/38.946629.
18. Marc Macenko et al. “A method for normalizing histology slides for quantitative analysis”. In: *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. 2009, pp. 1107–1110. doi: 10.1109/isbi.2009.5193250.
19. Abhishek Vahadane et al. “Structure-Preserving Color Normalization and Sparse Stain Separation for Histological Images”. In: *IEEE Transactions on Medical Imaging* 35.8 (2016), pp. 1962–1971. doi: 10.1109/tmi.2016.2529665.
20. Karel Zuiderveld. “Contrast Limited Adaptive Histogram Equalization”. In: *Graphics Gems IV*. Ed. by Paul S. Heckbert. Academic Press Professional, 1994, pp. 474–485. ISBN: 9780123361554. doi: 10.1016/b978-0-12-336156-1.50061-6.
21. David Hirling et al. “Segmentation metric misinterpretations in bioimage analysis”. In: *Nature Methods* 20 (2023), pp. 1256–1258. doi: 10.1038/s41592-023-01942-8.

22. Yuxin Wu et al. *Detectron2*. 2019. URL: <https://github.com/facebookresearch/detectron2>.
23. David A. Van Valen et al. “Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-Cell Imaging Experiments”. In: *PLOS Computational Biology* 12.11 (2016), e1005177. DOI: 10.1371/journal.pcbi.1005177.
24. Uwe Schmidt et al. “Cell Detection with Star-Convex Polygons”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2018*. Springer, 2018, pp. 265–273. DOI: 10.1007/978-3-030-00934-2_30. arXiv: 1806.03535.
25. Roboflow. *Supervision: Reusable Computer Vision Tools*. Open-source Python library for building computer vision applications. 2024. URL: <https://github.com/roboflow/supervision>.

Chapter 3

EQUIVARIANT MASK2FORMER FOR CELL SEGMENTATION

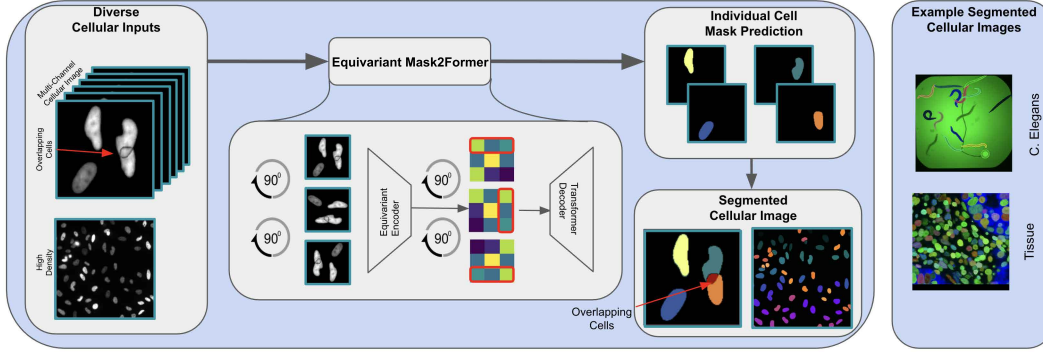


Figure 3.1: Overview of the Equivariant Mask2Former workflow. Cellular images are processed by a Mask2Former-based architecture with an equivariant feature-processing path and a standard transformer decoder that predicts individual cell masks and produces instance-segmented cellular images.

3.1 Introduction

Cell segmentation converts microscopy images into object-level measurements that can be counted, tracked, and related to molecular state. In many biological settings, however, the annotation budget is small. Labels may require expert review, cell boundaries can be ambiguous, and datasets often differ in modality, specimen type, staining strategy, resolution, and object density. A practical segmentation model should therefore learn efficiently from limited data while remaining flexible across nuclei, whole cells, bacteria, yeast, and other biological objects.

Microscopy also has a useful geometric structure. For many cell-segmentation problems, in-plane rotations and reflections change the coordinate frame of the image but not the identity of the biological object. A round nucleus, an elongated bacterium, or a worm can appear at many orientations, and the segmentation mask should rotate with the image. This is an equivariance requirement, not an invariance requirement. If f maps an image to a mask and $g \in G$ indexes a geometric transformation, segmentation ideally satisfies

$$f(T_g x) = T'_g f(x), \quad g \in G. \quad (3.1)$$

Here T_g is the operation associated with g on the input image x , while T'_g is the corresponding but distinct operation on the output prediction $f(x)$. The prime is important: T_g and T'_g are indexed by the same group element, but they act on different spaces or representations¹. Classification often seeks invariance, which can be viewed as the special case where T'_g is the identity; segmentation requires the output coordinates to co-vary with the input.

Standard convolution gives translation equivariance through weight sharing, but it does not give exact rotation or reflection equivariance. The usual workaround is data augmentation: show the network many rotated and flipped examples so that it learns approximate orientation robustness. Augmentation is useful, but it spends data and optimization capacity learning a regularity that can be encoded directly. Group-equivariant and steerable networks instead share filters across transformations, constraining feature maps to transform predictably under a chosen group^{2–5}. This chapter asks how far that prior can be carried into a modern transformer-based segmentation architecture.

Mask2Former is an attractive base model because it frames segmentation as query-based mask prediction rather than as dense per-pixel classification alone⁶. Object queries interact with multi-scale image features, and masked attention focuses each query on regions that are likely to belong to its predicted mask. This design is useful for cellular images because it can represent instances directly and can, in principle, separate touching or overlapping objects without hand-designing a different output representation for every cell type. The limitation is that a vanilla Mask2Former inherits the geometric behavior of its feature extractor and attention stack; it is not rotation-equivariant by construction.

This chapter studies Equivariant Mask2Former, a Mask2Former-based instance segmentation pipeline in which the ResNet encoder and FPN pixel decoder are replaced with finite rotation/reflection-equivariant modules built with *escnn*^{4,7}. The transformer decoder is retained as the standard Mask2Former masked-attention decoder. The goal is diagnostic: to test when explicit finite-group feature processing improves training under limited annotation, reduced augmentation, or missing pretraining, and when its computational cost outweighs its benefit. This question motivates Chapter 4, which moves from architectural priors toward detector-guided foundation-model adaptation. The chapter makes three contributions:

1. It adapts a Mask2Former-based instance segmentation pipeline by replacing

the image feature-processing path with finite rotation/reflection-equivariant *escnn* modules while retaining the query-based mask decoder.

2. It organizes the evaluation around controlled factors that matter for annotation-efficient microscopy: augmentation, pretraining, backbone equivariance, and pixel-decoder equivariance.
3. It frames equivariance as a cost-benefit tradeoff by pairing accuracy-oriented comparisons with runtime, memory, and limitation analysis.

3.2 Related Work

Cell and microscopy segmentation

Classical biomedical segmentation architectures such as U-Net established that convolutional encoder-decoder models can work with limited biomedical labels when combined with careful augmentation⁸. Later cell-specific systems incorporated stronger object priors. StarDist represents objects with star-convex polygons⁹; Cellpose uses flow-field representations for cellular masks^{10,11}; Mesmer targets whole-cell and nuclear segmentation in tissue images¹²; Omnipose extends flow-based segmentation to elongated bacterial morphologies¹³. These methods show that cell segmentation benefits from biological priors, but they also illustrate a recurring tradeoff: a representation tuned for one morphology or imaging regime may require adaptation for another.

The datasets used in this chapter follow this theme. They include TissueNet tissue segmentation and *C. elegans* images from the Broad Bioimage Benchmark Collection^{12,14}. These datasets test complementary regimes: TissueNet provides a larger tissue-cell dataset, whereas *C. elegans* provides a small-data setting with elongated and overlapping biological objects.

Transformer-based segmentation

Transformers introduced a general attention mechanism for sequence modeling¹⁵, and vision transformers adapted attention to images through patch-based representations¹⁶. Work has shown that they are capable of generalizing across a wide range of tasks and data modalities¹⁷. For imaging data, attention can capture both local and global relationships across length scales. Swin Transformers¹⁸ have proven effective at feature extraction from images at different resolutions and object densities. DETR showed that detection can be formulated as set prediction with learned object queries¹⁹. Mask2Former extends this family by using masked attention and a

unified segmentation interface for semantic, instance, and panoptic segmentation⁶. It offers several advantages over existing cell segmentation methods. First, instance segmentation is posed as a set prediction task, in principle allowing for the segmentation of overlapping objects. Second, the model uses query vectors to learn what counts as an object, so it can learn object-level representations directly instead of relying only on local boundary or distance-map cues.

Group equivariant networks

Group-equivariant CNNs extend convolution from translations to larger symmetry groups by sharing filters over transformed copies of a pattern². The steerable CNN framework generalizes this idea through group representations and constrained kernels, allowing feature fields to transform according to prescribed transformation laws^{3,4}. Harmonic networks provide another route by using circular harmonics to obtain translation and rotation equivariance⁵. In biomedical imaging, roto-translation equivariant networks and rotation-equivariant microscopy models have shown that explicit symmetry can improve robustness or reduce the amount of augmentation needed in orientation-variable image analysis^{20,21}.

Equivariant attention

Attention layers are not automatically equivariant, particularly when positional encodings and tokenization break the transformation structure of the image. Several works therefore modify attention with group-aware positional encodings or equivariant feature types, including the SE(3)-Transformer²², LieTransformer²³, group-equivariant self-attention²⁴, and $E(2)$ -equivariant vision transformers²⁵. These works motivate equivariant transformer decoders, but the implementation in this chapter is more conservative: equivariance is introduced into the convolutional feature hierarchy, while the Mask2Former decoder is treated as a non-equivariant consumer of equivariant or projected image features.

3.3 Method

This section introduces the implemented Equivariant Mask2Former architecture. The model retains the query-based mask decoder from Mask2Former⁶, but changes the image feature-processing path so that the ResNet encoder and FPN pixel decoder use finite rotation/reflection-equivariant modules. Cellular image data often have no canonical in-plane orientation, so this prior is a plausible way to reduce dependence on rotation and flip augmentation. The experimental question is not whether equiv-

ariance is always better, but when this inductive bias helps under limited labels, missing pretraining, or weak augmentation.

In the following section, we describe the construction of the equivariant architecture.

Architecture Overview

The model adopts an encoder-decoder architecture based on Mask2Former⁶. It has three parts. First, an image encoder projects the input image into a higher-dimensional feature map. Second, a pixel decoder upsamples and fuses multi-scale feature maps. Third, the standard Mask2Former transformer decoder processes the pixel-decoder features with object queries to predict cell masks. An overview is shown in Figure 3.2.

Designing Equivariant Image Encoders

Image encoders embed images into higher-dimensional feature representations. For visual tasks, a common design choice is ResNet²⁶, a convolutional architecture with residual connections. ResNet encoders have many parameters, making them sensitive to training-set size. When training data are limited, it is common to initialize image encoders from larger available corpora such as ImageNet^{6,19}. Cellular image data differ from natural images, and an ImageNet-scale corpus with matching microscopy modalities is not available for every biological segmentation task.

One solution is to train the image encoder from a random initialization. However, in the absence of large amounts of training data, this leads to significant overfitting and poor performance, as we show in our experiments. For the domain of cellular images, we propose equivariant image encoders. Equivariant image encoders exploit the underlying regularities that exist in data to increase the model’s statistical efficiency. A common geometric prior, or regularity, that exists in cellular images is symmetry. Specifically, translations, rotations, and reflections are isometries of Euclidean space. The full continuous symmetry group is the Euclidean group, $E(2)$, but the implemented model uses a finite rotation/reflection subgroup of this group. We therefore define the feature transformations with respect to the finite group used in the code rather than claiming equivariance to all of $E(2)$.

Definition of equivariance.

Mathematically, for a group G , equivariance is a property of an operator $f : X \rightarrow Y$ (e.g., a neural network layer) that commutes with the group action. Using the T_g/T'_g

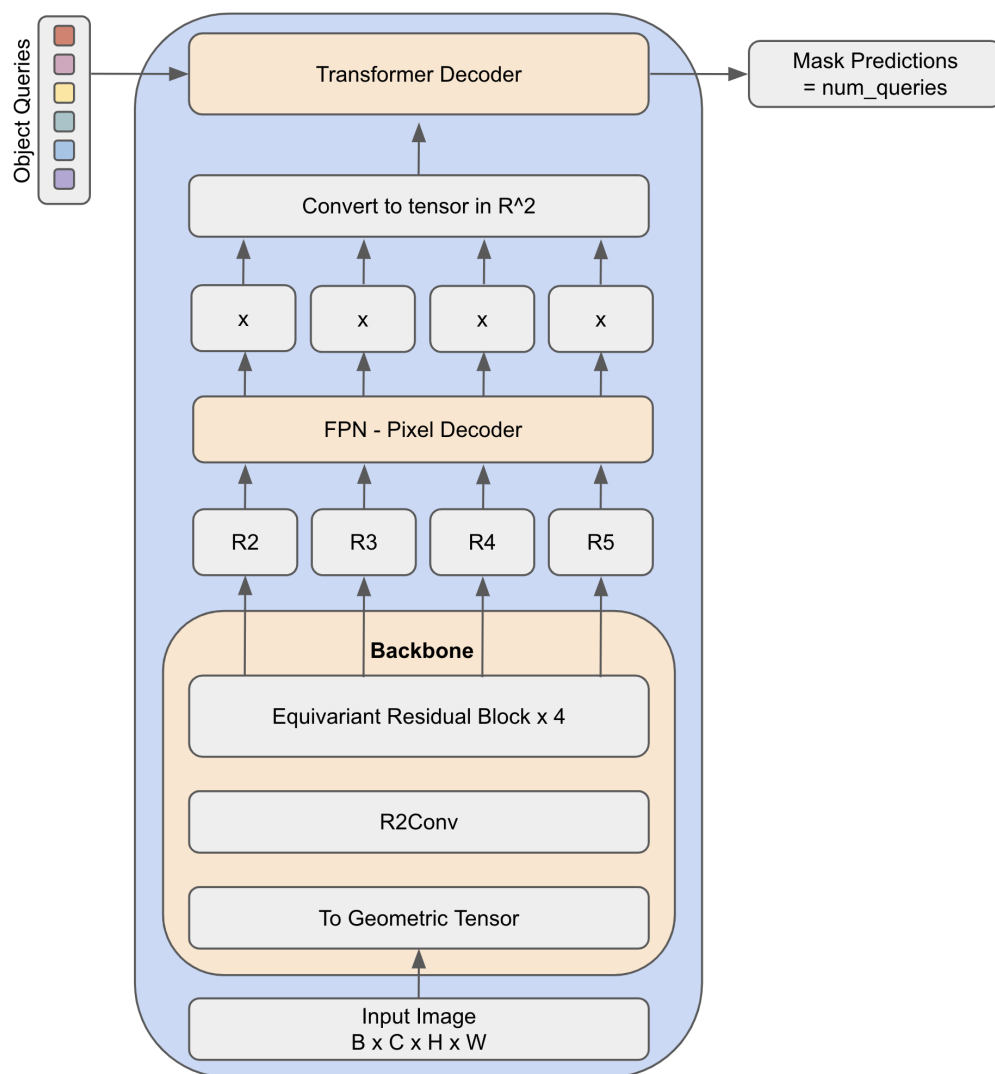


Figure 3.2: Architecture of Equivariant Mask2Former. The image encoder and pixel decoder produce multi-scale image features, and the masked-attention transformer decoder predicts cell-instance masks from object queries.

format, this means

$$f(T_g x) = T'_g f(x), \quad g \in G, x \in X. \quad (3.2)$$

Here $T_g : X \rightarrow X$ is the transformation applied to the input x , and $T'_g : Y \rightarrow Y$ is the corresponding but distinct transformation applied to the output or feature value $f(x)$. Thus g names the abstract group element, while T_g and T'_g specify how that same element is represented on the two spaces.

Finite rotation/reflection-equivariant convolutions.

A regular convolution of an input f with filter w , defined as

$$(w \circ f)(x) = \sum_y f(y)w(y - x) \quad (3.3)$$

is translation equivariant: if t_a shifts an image by displacement a , then

$$w \circ (t_a f) = t_a(w \circ f). \quad (3.4)$$

This property does not make a standard convolution equivariant to rotations or reflections. In this chapter, the additional symmetry group is the finite dihedral group implemented by *escnn*'s `flipRot2dOnR2(N=8)` gspace. This group can be written as $G = C_8 \rtimes C_2$: eight rotations by multiples of $2\pi/8$ together with a reflection, giving 16 group actions. The spatial translations are still handled by the convolutional structure, while the steerable layer controls how channels transform under G .

The formulation below follows the lifting, group-convolution, and projection view of roto-translation covariant CNNs²⁰, specialized here to the finite rotation/reflection group used by *escnn*. It is useful to separate the first lifting step from later group-equivariant convolutions. A lifting convolution maps a planar signal to a feature field indexed by position and a finite group element,

$$(L_w f)(x, g) = \sum_y f(y)w(g^{-1}(y - x)), \quad g \in G. \quad (3.5)$$

For rotations alone, g may be denoted by an angle θ , yielding the familiar expression $(L_w f)(x, \theta) = \sum_y f(y)w(R_\theta^{-1}(y - x))$. In our implementation, however, g also includes the reflection element, so the output domain is $\mathbb{Z}^2 \times G$, not only $\mathbb{Z}^2 \times \Theta$. Subsequent steerable convolutions map between feature fields whose channels carry specified representations of G ; they are not merely ordinary convolutions over an angular index.

After lifting, a group feature can be written as $F : \mathbb{Z}^2 \times G \rightarrow \mathbb{R}^{c_{\text{in}}}$. With the same correlation convention used above, a group convolution with kernel K produces $F' : \mathbb{Z}^2 \times G \rightarrow \mathbb{R}^{c_{\text{out}}}$ by

$$F'(x, h) = \sum_{y \in \mathbb{Z}^2} \sum_{g \in G} K(h^{-1}(y - x), h^{-1}g) F(y, g), \quad h \in G. \quad (3.6)$$

The kernel is therefore indexed by a spatial offset expressed in the output frame and by the relative group element between the input and output feature types. For the rotation-only example $G = C_8 = \{R_j\}_{j=0}^7$, this becomes

$$F'(x, R_j) = \sum_{y \in \mathbb{Z}^2} \sum_{i=0}^7 K(R_{-j}(y - x), R_{i-j}) F(y, R_i), \quad (3.7)$$

with orientation indices taken modulo 8. Thus the output slice at orientation R_j compares every input orientation R_i to a kernel slice at the relative orientation R_{i-j} , while rotating the spatial offset into the R_j output frame. The reflection case used by `flipRot2dOnR2(N=8)` follows the same formula with $G = C_8 \rtimes C_2$ and 16 group elements instead of eight rotations.

To return a group-indexed feature field to an ordinary feature map on $\mathbb{R}^2$, one can pool over the fiber G at each spatial location. Average and max fiber pooling are

$$(P_{\text{avg}} F)(x) = \frac{1}{|G|} \sum_{g \in G} F(x, g), \quad (P_{\text{max}} F)(x) = \max_{g \in G} F(x, g), \quad (3.8)$$

applied channel-wise. In the rotation-only C_8 case, $P_{\text{avg}} F(x) = \frac{1}{8} \sum_{j=0}^7 F(x, R_j)$ and $P_{\text{max}} F(x) = \max_j F(x, R_j)$. If the input is rotated by R_k , the group action rotates the spatial coordinate and cyclically permutes the orientation index; summing or maximizing over all orientations removes this index, giving $P(T_{R_k} F)(x) = P(F)(R_k^{-1}x)$. Fiber pooling therefore maps group features back to $\mathbb{R}^2$ scalar-field features: the feature map still moves spatially with the image, but it no longer carries an explicit orientation or reflection channel.

Constructing an equivariant convolution.

We describe how a convolutional layer in a CNN or a ResNet can be made equivariant. CNNs succeed in computer vision in large part because they have translational equivariance (shifts in the inputs lead to shifts in the outputs) built in. The theory of *Steerable CNNs*²⁻⁴ provides a framework for extending the translational equivariance that exists in CNNs to other transformations such as rotations and reflections.

We use the *escnn* PyTorch library^{4,7} to create convolutional layers equivariant to the finite group $G = C_8 \rtimes C_2$. The library specifies the *gspace*, the input and output *FieldTypes*, and the representation carried by each feature field. It also supports lifting from image-domain scalar fields to regular-representation feature fields and pooling or projection back to ordinary image-domain tensors when needed.

Integrating Equivariance into Mask2Former Architecture

We apply finite rotation/reflection equivariance to the feature-processing path, which requires altering two components of the Mask2Former architecture: the backbone (ResNet-50) and the pixel decoder (Feature Pyramid Network, FPN). We use the PyTorch package *escnn*^{4,7} to rebuild both of these networks. This library provides a way of defining common neural network components (e.g., convolutions, non-linearities, and normalizations) with respect to a specific symmetry group acting on $\mathbb{R}^2$. In the following section, we describe the specific changes made to the ResNet-50 and FPN to make them equivariant. We keep the original transformer mask decoder; it receives the processed image features and retains the standard Mask2Former query-decoding interface.

The implementation uses the *escnn*^{4,7} package to define feature fields and equivariant layers. The *GSpace* class specifies the symmetries with respect to which the network should be equivariant. In this chapter, we use the *flipRot2dOnR2(N=8)* *gspace*. This represents eight discrete rotations by multiples of $2\pi/8$ and a flip action, giving 16 distinct transformations (or group actions). To make the network equivariant with respect to these transformations, we specify a deterministic transformation law for the feature space induced by transformations of the input space. This is achieved using the *R2Conv* class, which creates G-steerable planar convolutions mapping between input and output *FieldTypes*. A *FieldType* describes both the base space on which a feature field lives and the representation of the chosen group carried by its channels. Given this transformation law, *R2Conv* defines convolutions equivariant to the finite rotation/reflection group. Instead of providing only the number of input and output channels to *R2Conv*, we specify the input *FieldType* and output *FieldType* with their multiplicities.

Equivariant ResNet and FPN.

We base our ResNet on the Detectron2²⁷ implementation. The first equivariant submodule, the stem, maps an RGB input image to a feature space with a specified

FieldType and multiplicity (channels). To maintain a similar number of parameters and dimensionality as the original ResNet implementation, the three color channels of an RGB image are identified as three independent scalar fields, which transform under the trivial representation (with multiplicity equal to 3) to an output of 4 feature fields of regular representations (with multiplicity 4). This maps RGB (3 channels) to feature space (64 channels) because the regular representation with $N=8$ discrete rotations multiplied by 2 for flipping about the x-axis (i.e., $(8*2)*4 = 64$ channels). Note that these implementation details are about reconstructing the functionality of the standard ResNet and that these multiplicities can be changed for alternative input dimensions (non-RGB images). After the stem, the features pass through a sequence of stages composed of skip connections, convolutions, normalization, and activations. These can all be defined with the *escnn* package by specifying the FieldTypes. Because we are using *flipRot2dOnR2(N=8)*, the regular representation has 16 channels, so we can match the channel widths of a standard ResNet-50 by choosing multiplicity $m = \frac{c}{16}$ for target width c . Thus, by connecting m feature fields of the finite group regular representation, we can construct outputs at the usual ResNet-50 levels: 256 ($m = 16$), 512 ($m = 32$), 1024 ($m = 64$), and 2048 ($m = 128$).

To construct the FPN we use the Mask2Former implementation as our base. Similar to the ResNet-50 we replace all of the layers with their equivariant counterparts while preserving the dimensionality by making our representation multiplicities a function of the number of channels. The FPN requires one additional component: upsampling. For this, we use *R2Upsampling*, which interpolates while preserving equivariance.

3.4 Experiments

This section evaluates finite rotation/reflection-equivariant feature processing on biological datasets. First, it tests whether equivariance improves learning on a small *C. elegans* dataset when models are trained from scratch or without augmentation. It then evaluates the same design on the larger TissueNet dataset, where the benefit of explicit equivariance is less clear and computational cost becomes more important.

Datasets

This section describes the datasets used in the experiments. Table 3.1 summarizes the number of images and annotations in the training and validation sets.

TissueNet¹² (version 1.1) is a multiplexed tissue dataset for training segmentation

Dataset	Train Images	Train Annotations	Val Images	Val Annotations
TissueNet (Nuclear)	2576	929,959	3105	269,657
TissueNet (whole cell)	2576	983,470	3105	288,022
<i>C. elegans</i>	80	1121	20	286

Table 3.1: Overview of datasets for Equivariant Mask2Former.

models and contains more than 1 million manually labeled cells from different tissue types. Each TissueNet image contains two channels: a nuclear channel, such as DAPI, and a membrane or cytoplasmic channel, such as E-Cadherin or Pan-Keratin. The dataset used here contains 2576 512×512 images in the training set and 3105 256×256 images in the validation set.

Preprocessing: The nuclear target is selected for the segmentation task. The two-channel images are converted to RGB by mapping the nuclear and membrane channels to the green and blue channels, respectively, normalizing each channel to the range 0–1, and leaving the red channel empty. Training images are then tiled to 256×256 pixels with 50% overlap, producing 23,184 RGB training images. The maximum annotation count per cropped image is 523 for nuclei and 525 for whole cells (image 10354006 in the cropped training set).

*C. elegans*¹⁴ contains 100 16-bit 696×520 images of the roundworm *C. elegans*. Each image contains bright-field (BF) and GFP channels. The dataset includes overlapping worms, which are difficult objects for per-pixel segmentation models.

Preprocessing: The 100 images are split into 80 training images and 20 validation images. Only the BF channel is kept because the GFP channel is used for identifying dead worms, not for segmentation. The central 512×512 crop is placed in the green channel of an RGB image.

Training and evaluation protocol

C. elegans training without augmentation

The *C. elegans* dataset is processed into COCO format using the reusable preprocessing library described in Chapter 2. Unless otherwise stated, this experiment uses no rotation or flip augmentation. Models are trained with a batch size of 40 on two A6000 GPUs, using an input image size of 512×512 pixels for 100k iterations. Optimization uses AdamW with a learning rate of 1×10^{-5} for the backbone and

1×10^{-4} for the remaining model components. Performance is evaluated with COCO AP and AP50.

The comparison uses three setup families: a regular ResNet/FPN baseline, models with both equivariant encoder and equivariant FPN, and mixed regular/equivariant feature-processing variants. The display names are:

1. R50+FPN pretrained on ImageNet.
2. R50+FPN trained from scratch.
3. eR50+eFPN trained from scratch.
4. eR50+FPN trained from scratch.
5. R50+eFPN pretrained on ImageNet.

In these names, R50 denotes a regular ResNet-50 backbone, FPN denotes a regular feature pyramid network, eR50 denotes an equivariant ResNet-50 backbone, and eFPN denotes an equivariant feature pyramid network.

C. *elegans* training with augmentation

The augmented *C. elegans* experiment uses the same setup as the no-augmentation setup. The only difference is that random rotation, random flipping, and color augmentations are applied during training. The same five model configurations listed above are evaluated under this augmented training setting.

TissueNet training

The TissueNet dataset is processed into COCO format using the reusable preprocessing library described in Chapter 2. The model uses `num_dec_layer=4`, `nheads=1`, `img_size=256`, and `num_queries=400`. Training runs for 200k iterations with a batch size of 8 on one A6000 GPU. No augmentation is applied. Optimization uses AdamW with a learning rate of 1×10^{-5} for the backbone and 1×10^{-4} for the remaining model components. Performance is evaluated with COCO AP and AP50. Two models are compared: a regular model from scratch and a fully equivariant model from scratch.

3.5 Results

C. elegans training without augmentation

Removing augmentation creates a direct stress test of the equivariance prior. As shown in Figure 3.3 and Table 3.2, the eR50+eFPN model trained from scratch performs best in this setting, reaching 30.3 AP and 58.4 AP50. The mixed eR50+FPN model trained from scratch is second, while the ImageNet-pretrained regular R50+FPN baseline drops to 6.63 AP and 17.9 AP50. The regular R50+FPN model trained from scratch is unstable. These results indicate that, when random rotations and flips are absent, explicitly equivariant feature processing can provide a stronger geometric prior than relying on pretraining alone.

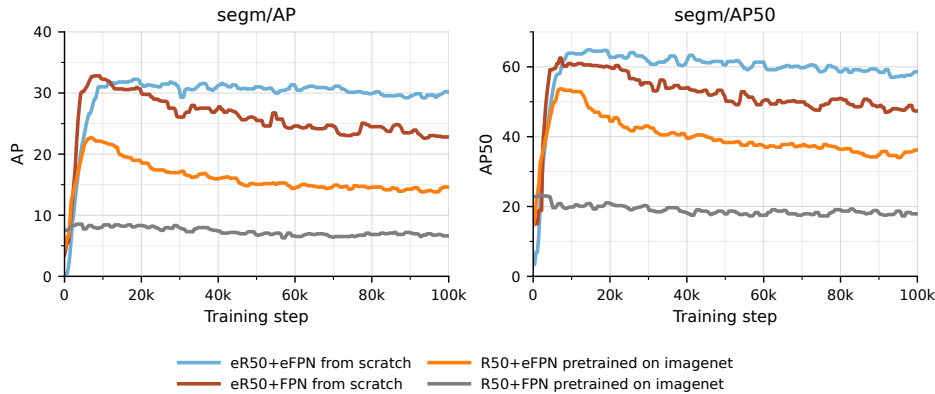


Figure 3.3: Evaluation curves for COCO AP and AP50 on the *C. elegans* training-without-augmentation experiment.

C. elegans training with augmentation

The augmented small-data experiment compares the same regular, mixed, and equivariant feature-processing configurations. Figure 3.4 shows the AP and AP50 evaluation curves, and Table 3.2 summarizes the final scores with and without augmentation. Adding augmentation improves the pretrained and equivariant-from-scratch configurations relative to the no-augmentation setting, with the largest AP increase for the ImageNet-pretrained regular baseline. However, augmentation does not rescue the regular R50+FPN model trained from scratch, which remains unstable. The ImageNet-pretrained regular baseline has the highest AP under augmentation, while the equivariant models remain trainable from scratch and achieve usable AP/AP50 without pretraining.

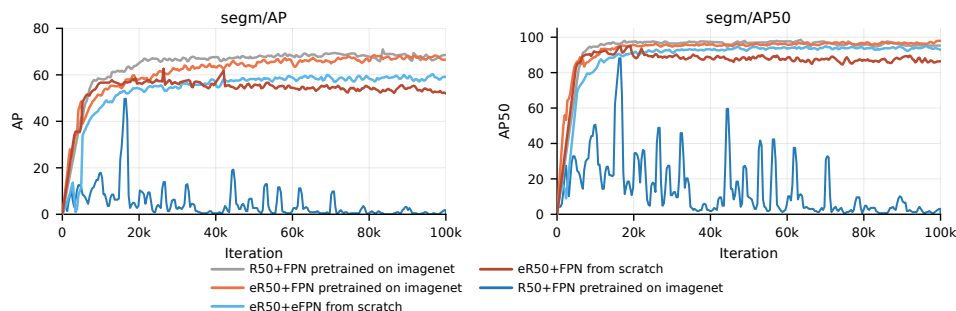


Figure 3.4: Evaluation curves for COCO AP and AP50 on the *C. elegans* training-with-augmentation experiment.

Model	COCO AP		COCO AP50	
	No aug.	Aug.	No aug.	Aug.
R50+FPN pretrained on ImageNet	6.6292	68.5102	17.8933	95.2697
R50+eFPN pretrained on ImageNet	14.6067	66.4956	36.1798	97.9474
eR50+eFPN from scratch	30.3147	59.0486	58.3955	93.7095
eR50+FPN from scratch	22.8090	51.9454	47.3385	86.3439
R50+FPN from scratch	unstable	unstable	unstable	unstable

Table 3.2: *C. elegans* instance segmentation results with and without augmentation. Entries marked *unstable* denote a training run that diverged: the loss did not converge and evaluation metrics fluctuated before collapsing to approximately zero, so no usable checkpoint could be selected.

TissueNet training

TissueNet provides a larger-data test of the same equivariance prior. Figure 3.5 shows that the equivariant model had not fully saturated by the end of training, so longer training could plausibly improve its final score. Even so, Table 3.3 indicates that equivariance did not improve COCO AP or AP50 in this larger-dataset regime under the current training schedule: the regular R50+FPN model trained from scratch reached 46.7 AP and 83.8 AP50, compared with 43.9 AP and 79.9 AP50 for the equivariant model. This result contrasts with the small-data experiments and suggests that, once enough labeled data are available, a standard R50 model can learn strong segmentation features from scratch without requiring an explicit rotation/reflection prior.

Qualitative observations

Figures 3.6 and 3.7 show representative qualitative predictions from the *C. elegans* and TissueNet experiments. The *C. elegans* examples illustrate the practical meaning

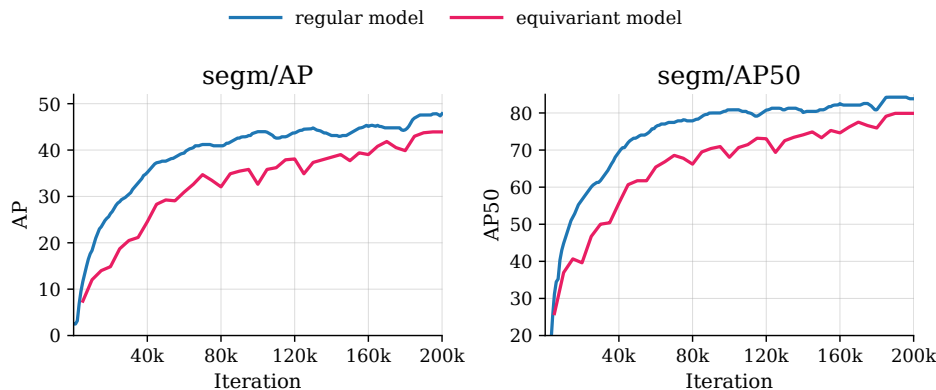


Figure 3.5: Evaluation curves for COCO AP and AP50 on the TissueNet training without augmentation experiment.

Model	COCO AP	COCO AP50
Regular model	46.7125	83.8298
Equivariant model	43.9157	79.8969

Table 3.3: TissueNet instance segmentation results without augmentation.

of the AP values: results with AP below 10 contain many unusable predictions, AP values around 30 show partial instance recovery but remain insufficient for reliable analysis, and AP values around 60 show substantially better separation. Some circular-worm examples contain artifacts because the polygon encoding used in this workflow does not support masks with holes. In contrast, the TissueNet examples are visually similar across models, consistent with the close quantitative metrics reported in Table 3.3.

Computational cost

Equivariant feature maps carry extra group dimensions or representation channels, so the expected cost is higher memory and runtime. Table 3.4 reports the observed TissueNet training cost for the regular and equivariant models. The equivariant model is substantially more expensive, increasing step time from 0.56 s to 3.62 s and peak GPU memory from 3.4 GB to 26.3 GB. In this TissueNet run, that additional cost does not translate into higher AP: the regular model reaches 46.7 AP, while the equivariant model reaches 43.9 AP. This result highlights an important practical tradeoff. The equivariant prior can be useful in low-data or no-augmentation settings, but it must be weighed against the large runtime and memory overhead when training

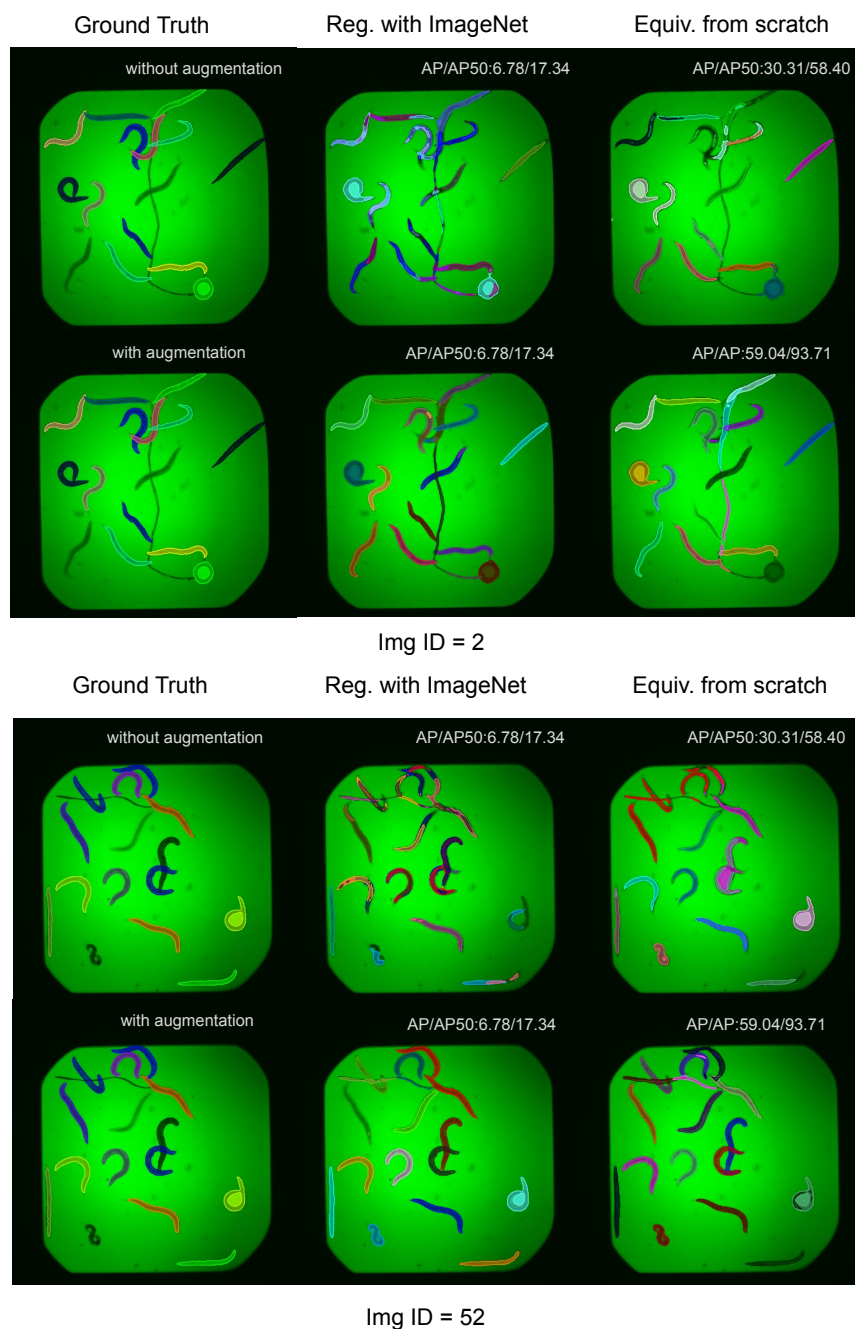
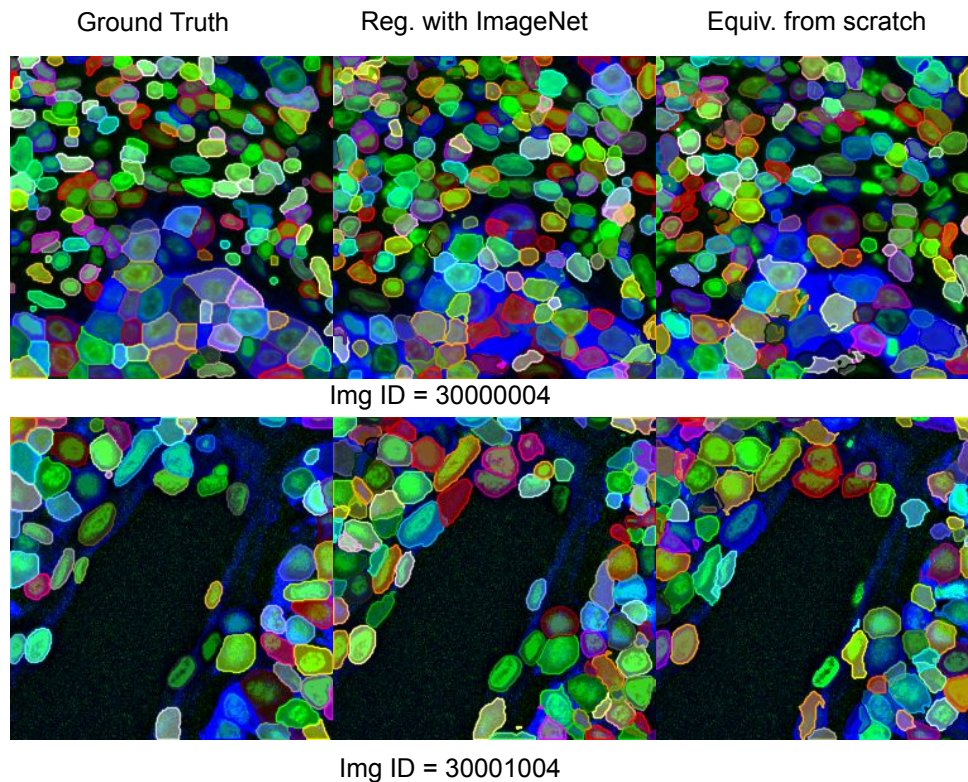


Figure 3.6: Representative qualitative segmentation visualizations for the *C. elegans* experiments. AP below 10 corresponds to poor predictions, AP around 30 gives only modest improvement and remains unsuitable for reliable analysis, and AP around 60 produces better instance masks. Circular worms can show artifacts because the polygon encoding does not support masks with holes.



Tissuenet with augmentation,
 Regular model. AP / AP50= 46.7 / 83.8
 equivariant model. AP / AP50= 44.1 / 85.2

Figure 3.7: Representative qualitative segmentation visualizations for the TissueNet experiments. The regular and equivariant models produce visually similar predictions, matching the relatively close AP and AP50 values in the quantitative comparison.

on larger or denser datasets.

Metric	Regular model	Equivariant model
Step time (s)	0.56	3.62
GPU memory (GB)	3.4	26.3
COCO AP	46.7	43.9

Table 3.4: TissueNet training cost and AP comparison for regular and equivariant models.

3.6 Discussion

Equivariant Mask2Former tests an old but important idea in a modern segmentation setting: if the data distribution has a known symmetry, the model should not have

to rediscover it from labels. For cells and many microscopy objects, arbitrary orientation is often a nuisance factor. Equivariant encoders and pixel decoders can turn that nuisance into weight sharing. This prior is most useful when:

- Annotations are limited, so the model cannot easily learn orientation robustness from examples alone;
- Rotation and flip augmentation is unavailable, too costly, or insufficient for the target domain; and
- Pretrained weights are unavailable, or pretraining a large backbone model is beyond the compute budget.

The same inductive bias has limits. First, random rotation and flip augmentation can be a simpler and cheaper way to handle orientation variation, and in some settings may outperform the more complex implementation of strict equivariance. Second, enforcing equivariance through representation fields increases implementation complexity and computational cost. Third, the implemented equivariance remains approximate: striding and padding can break exact transformation consistency, and square convolutional kernels are only exactly compatible with rotations by multiples of 90° unless the filter basis and sampling scheme are designed to support the chosen rotation group.

The broader thesis lesson is that architectural priors and data scale solve related problems from opposite directions. Equivariant Mask2Former asks whether a carefully chosen geometric prior can make a segmentation model more data-efficient. Chapter 4 asks whether broad pretraining and a generalist segmentation interface can reduce the need for hand-designed priors. The transition is natural: when labels are scarce and the symmetry is reliable, equivariance can be a strong specialist bias; when training data and model scale grow, a foundation/generalist model may learn a wider range of visual variation, including cases where strict symmetry is too narrow.

References

1. Taco Cohen and Max Welling. “Group Equivariant Convolutional Networks”. In: *Proceedings of the 33rd International Conference on Machine Learning*. PMLR, 2016, pp. 2990–2999. arXiv: 1602.07576.
2. Taco S. Cohen and Max Welling. “Steerable CNNs”. In: *arXiv preprint* (2016). arXiv: 1612.08498.

3. Taco S. Cohen, Mario Geiger, and Maurice Weiler. “A General Theory of Equivariant CNNs on Homogeneous Spaces”. In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019. arXiv: 1811.02017.
4. Maurice Weiler and Gabriele Cesa. “General E(2)-Equivariant Steerable CNNs”. In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019. arXiv: 1911.08251.
5. Daniel E. Worrall et al. “Harmonic Networks: Deep Translation and Rotation Equivariance”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5028–5037. arXiv: 1612.04642.
6. Bowen Cheng et al. “Masked-Attention Mask Transformer for Universal Image Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1290–1299. doi: 10.1109/cvpr52688.2022.00135. arXiv: 2112.01527.
7. Gabriele Cesa, Leon Lang, and Maurice Weiler. “A Program to Build E(N)-Equivariant Steerable CNNs”. In: *International Conference on Learning Representations*. 2022. URL: <https://openreview.net/forum?id=WE4qe9xlnQw>.
8. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer, 2015, pp. 234–241. doi: 10.1007/978-3-319-24574-4_28. arXiv: 1505.04597.
9. Uwe Schmidt et al. “Cell Detection with Star-Convex Polygons”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2018*. Springer, 2018, pp. 265–273. doi: 10.1007/978-3-030-00934-2_30. arXiv: 1806.03535.
10. Carsen Stringer et al. “Cellpose: a generalist algorithm for cellular segmentation”. In: *Nature Methods* 18.1 (Jan. 2021), pp. 100–106. doi: 10.1038/s41592-020-01018-x.
11. Marius Pachitariu and Carsen Stringer. “Cellpose 2.0: how to train your own model”. In: *Nature Methods* 19.12 (Dec. 2022), pp. 1634–1641. doi: 10.1038/s41592-022-01663-4.
12. Noah F. Greenwald et al. “Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning”. In: *Nature Biotechnology* 40.4 (Apr. 2022), pp. 555–565. doi: 10.1038/s41587-021-01094-0.
13. Kevin J. Cutler et al. “Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation”. In: *Nature Methods* 19.11 (Nov. 2022), pp. 1438–1448. doi: 10.1038/s41592-022-01639-4.

14. Vebjorn Ljosa, Katherine L. Sokolnicki, and Anne E. Carpenter. “Annotated high-throughput microscopy image sets for validation”. In: *Nature Methods* 9.7 (2012), p. 637. DOI: 10.1038/nmeth.2083.
15. Ashish Vaswani et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008. URL: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>.
16. Alexey Dosovitskiy et al. “An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations*. 2021. arXiv: 2010.11929.
17. Andrew Jaegle et al. “Perceiver IO: A General Architecture for Structured Inputs and Outputs”. In: *arXiv preprint* (2021). arXiv: 2107.14795.
18. Ze Liu et al. “Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022. DOI: 10.1109/iccv48922.2021.00986. arXiv: 2103.14030.
19. Nicolas Carion et al. “End-to-End Object Detection with Transformers”. In: *Computer Vision – ECCV 2020*. Springer, 2020, pp. 213–229. DOI: 10.1007/978-3-030-58452-8_13. arXiv: 2005.12872.
20. Erik J. Bekkers et al. “Roto-Translation Covariant Convolutional Networks for Medical Image Analysis”. In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*. Springer, 2018, pp. 440–448. DOI: 10.1007/978-3-030-00928-1_50. arXiv: 1804.03393.
21. Benjamin Chidester et al. “Rotation Equivariant and Invariant Neural Networks for Microscopy Image Analysis”. In: *Bioinformatics* 35.14 (2019), pp. i530–i537. DOI: 10.1093/bioinformatics/btz353.
22. Fabian B. Fuchs et al. “SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 1970–1981. arXiv: 2006.10503.
23. Michael J. Hutchinson et al. “LieTransformer: Equivariant Self-Attention for Lie Groups”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 4533–4543. arXiv: 2012.10885.
24. David W. Romero and Jean-Baptiste Cordonnier. “Group Equivariant Stand-Alone Self-Attention for Vision”. In: *International Conference on Learning Representations*. 2021. arXiv: 2010.00977.
25. Renjun Xu et al. “E(2)-Equivariant Vision Transformer”. In: *arXiv preprint* (2023). arXiv: 2306.06722.

26. Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778. DOI: 10.1109/cvpr.2016.90. arXiv: 1512.03385.
27. Yuxin Wu et al. *Detectron2*. 2019. URL: <https://github.com/facebookresearch/detectron2>.

Chapter 4

CELLSAM: A FOUNDATION MODEL FOR CELL SEGMENTATION ACROSS MODALITIES

This chapter is mainly based on the following:

1. Markus Marks et al. “CellSAM: a foundation model for cell segmentation”. In: *Nature Methods* 22.12 (Dec. 8, 2025), pp. 2585–2593. doi: 10.1038/s41592-025-02879-w.

Chapter overview. Cells are a fundamental unit of biological organization, and identifying them in imaging data—cell segmentation—is a critical task for cellular imaging experiments. This chapter presents CellSAM as a generalist model for automated cell segmentation across microscopy modalities. CellSAM builds on SAM by adding CellFinder, an automated detector that generates bounding-box prompts for dense cellular images, and then adapts the SAM-based mask-decoding workflow to cell segmentation. This chapter sits at the center of the thesis arc: after Chapter 2 establishes controlled preprocessing and evaluation, and Chapter 3 studies finite rotation/reflection-equivariant modeling under limited labels, CellSAM asks whether detector-guided prompting can move cell segmentation from a collection of specialist tools toward a generalist foundation-model interface. The chapter is based on the published CellSAM manuscript; Qilin Li’s contributions include algorithm design for CellFinder and CellSAM, CellFinder implementation, experiments and evaluations, model benchmarking, data pipelines and data engineering, and manuscript writing.

4.1 Introduction

Accurate cell segmentation is crucial for quantitative analysis and interpretation of various cellular imaging experiments. Modern spatial genomics assays can produce data on the location and abundance of 10^2 protein species and 10^3 RNA species simultaneously in living and fixed tissues^{1–5}. Accurate cell segmentation allows this type of data to be converted into interpretable tissue maps of protein localization and transcript abundances; these maps provide important insights into the biology of healthy and diseased tissues. Similarly, live-cell imaging provides insight into dynamic phenomena in bacterial and mammalian cell biology. Studying live-cell

imaging data has provided mechanistic insights into critical phenomena such as the mechanical behavior of the bacterial cell wall^{6,7}, information transmission in cell signaling pathways^{8–13}, heterogeneity in immune cell behavior during immunotherapy¹⁴, and the morphodynamics of development¹⁵. Cell segmentation is also a key challenge for these experiments, as cells must be segmented and tracked to create temporally consistent records of cell behavior that can be queried at scale. These methods have seen use in several systems, including mammalian cells in cell culture^{13,16} and tissues⁵, bacterial cells^{17–20}, and yeast^{21–23}.

Considerable progress has been made in recent years on the problem of cell segmentation, driven primarily by advances in deep learning²⁴. Progress in this space has occurred mainly in two distinct directions. The first direction seeks to find deep learning architectures that achieve state-of-the-art performance on cellular imaging tasks. These methods have historically focused on a particular imaging modality (for example, brightfield imaging) or target (for example, mammalian tissue) and have difficulty generalizing beyond their intended domain^{25–31}. For example, Mesmer’s²⁸ representation for a cell (cell centroid and boundary) enables good performance in tissue images but would be a poor choice for elongated bacterial cells. Similar trade-offs in representations exist for the current collection of Cellpose models, necessitating the creation of a model zoo^{26,32}. The second direction is to work on improving labeling methodology. Cell segmentation is an application of the instance segmentation problem, which requires pixel-level labels for every object in an image. Creating these labels can be expensive (10^{-2} USD/label, with hundreds to thousands of labels per image)^{28,33}, which provides an incentive to reduce the marginal cost of labeling. A recent improvement to labeling methodology has been human-in-the-loop labeling, where labelers correct model errors rather than produce labels from scratch^{26,28,34}. Further reductions in labeling costs can increase the amount of labeled imaging data by orders of magnitude.

Recent work in machine learning on foundation models holds promise for providing a complete solution. Foundation models are large deep neural network models (typically transformers³⁵) trained on large amounts of data in a self-supervised fashion with supervised fine-tuning on one or several tasks³⁶. Foundation models include the GPT^{37,38} family of models, which have proven transformative for natural language processing³⁶. These types of attention-based models have recently been used for processing biological sequences^{39–43}. These successes have inspired similar efforts in computer vision. The Vision Transformer (ViT)⁴⁴ was introduced in 2020

and has since been used as the basis architecture for a collection of vision foundation models^{45–49}. A key feature of foundation models is the scaling of model performance with model size, dataset size, and compute⁵⁰; these scaling laws have been observed for both language and vision models⁵¹. These scaling laws offer a path toward generalist models for cellular image analysis by increasing dataset and model size in exchange for dealing with the increased compute cost of training foundation models. This is in contrast to prior efforts that have focused on model architecture design and representation engineering.

One recent foundation model well suited to cellular image analysis is the Segment Anything Model (SAM)⁵². This model uses a ViT to extract information-rich features from raw images. These features are then directed to a module that generates instance masks based on user-provided prompts, which can be either spatial (e.g., an object centroid or bounding box) or semantic (e.g., an object’s visual description). Notably, the promptable nature of SAM enabled scalable dataset construction, as preliminary versions of SAM allowed labelers to generate accurate instance masks with one to two clicks. The final version of SAM was trained on a dataset of 11 million images containing over 1 billion masks and demonstrated strong performance on various zero-shot evaluation tasks. Recent work has attempted to apply SAM to problems in biological and medical imaging, including medical image segmentation^{53–56}, lesion detection in dermatological images^{57,58}, nuclear segmentation in H&E images^{59,60}, and cellular image data for use in the Napari software package⁶¹.

Works like MicroSAM⁶¹ or MedSAM⁵⁶ use SAM’s original workflow to speed up annotation of cells and medical data, label a large dataset, and then fine-tune the original SAM model. However, reliable automated segmentation is still missing in these works. While promising, these studies reported challenges adapting SAM to these new use cases^{53,61}. These challenges include reduced performance and uncertain boundaries when transitioning from natural to medical images. Cellular images contain additional complications: they can involve different imaging modalities (for example, phase microscopy versus fluorescence microscopy), thousands of objects in a field of view (as opposed to dozens in a natural image), and uncertain and noisy boundaries (artifacts of projecting three-dimensional objects into a two-dimensional plane)⁶¹.

In addition to these challenges, SAM’s default strategy for automatic prompting does not allow for accurate inference on cellular images. SAM’s automated prompting uses a uniform grid of points to generate masks, an approach poorly suited to cellular

images given the wide variation of cell densities. More precise prompting (e.g., a bounding box or mask) requires prior knowledge of cell locations. Because cellular images often contain a large number of cells, it is impractical for users to provide prompts to SAM manually. This limitation makes it challenging for SAM to serve as a foundation model for cell segmentation since it still requires substantial human input for inference. A solution that enables the automatic generation of prompts would enable SAM-like models to serve as foundation models and knowledge engines, as they could accelerate the generation of labeled data, learn from them, and make that knowledge accessible to life scientists via inference.

In this work, we developed CellSAM, a foundation model for cell segmentation (Fig. 4.1). CellSAM extends the SAM methodology to perform automated cellular instance segmentation. To achieve this, we first assembled a comprehensive dataset for cell segmentation spanning five broad data archetypes: tissue, cell culture, yeast, H&E, and bacteria. Critically, we removed data leaks between training and testing data splits to ensure an accurate assessment of model performance. To automate inference with CellSAM, we developed CellFinder, a transformer-based object detector that uses the Anchor DETR framework⁶². CellSAM and CellFinder share SAM’s ViT backbone for feature extraction; these features are first used by CellFinder to generate bounding boxes around the cells to be used as prompts for SAM. The bounding boxes (prompts) and ViT features are fed into a decoder to generate instance segmentations of the cells in an image. We trained CellSAM on a large, diverse corpus of cellular imaging data, enabling it to achieve state-of-the-art performance across ten datasets. We also evaluated CellSAM’s zero-shot performance using a held-out dataset, LIVECell⁶³, demonstrating that it substantially outperforms existing methods for zero-shot segmentation. A deployed version of CellSAM is available at <https://cellsam.deepcell.org>.

4.2 Results

Construction of a dataset for a generalist cell segmentation model

A significant challenge with existing cellular segmentation methods is their inability to generalize across cellular targets, imaging modalities, and cell morphologies. To address this, we curated a dataset from the literature containing 2D images from a diverse range of targets (mammalian cells in tissues and adherent cell culture, yeast cells, bacterial cells, and mammalian cell nuclei) and imaging modalities (fluorescence, brightfield, phase contrast, and mass cytometry imaging).

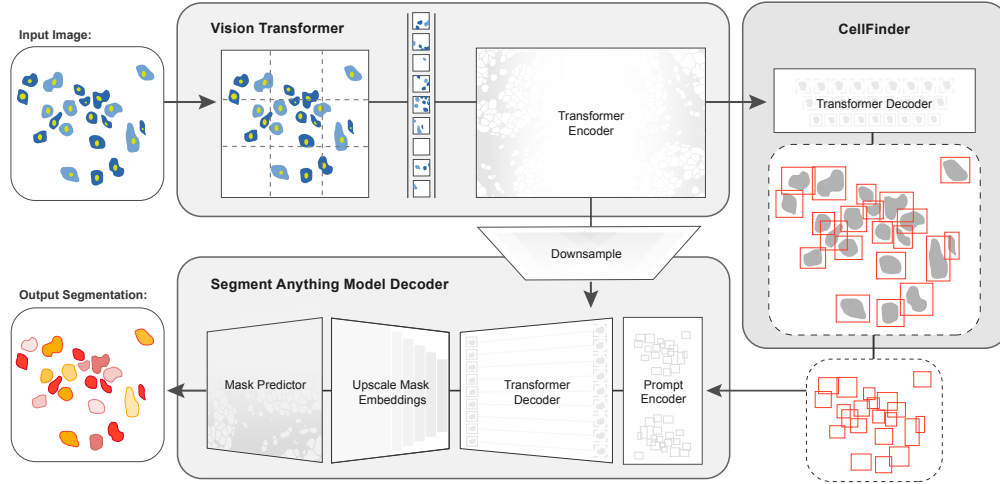


Figure 4.1: **CellSAM: a foundational model for cell segmentation.** CellSAM combines SAM’s mask generation and labeling capabilities with an object detection model to achieve automated inference. Input images are divided into regularly sampled patches and passed through a transformer encoder (i.e., a ViT) to generate information-rich image features. These image features are then sent to two downstream modules. The first module, CellFinder, decodes these features into bounding boxes using a transformer-based encoder-decoder pair. The second module combines these image features with prompts to generate masks using SAM’s mask decoder. CellSAM integrates these two modules using the bounding boxes generated by CellFinder as prompts for SAM. CellSAM is trained in two stages, using the pre-trained SAM model weights as a starting point. In the first stage, we train the ViT and the CellFinder model together on the object detection task. This yields an accurate CellFinder but results in a distribution shift between the ViT and SAM’s mask decoder. The second stage closes this gap by fixing the ViT and SAM mask decoder weights and fine-tuning the remainder of the SAM model (i.e., the model neck) using ground truth bounding boxes and segmentation labels.

Our final dataset consisted of TissueNet²⁸, DeepBacs⁶⁴, BriFiSeg⁶⁵, Cellpose^{25,26}, Omnipose⁶⁶, YeastNet⁶⁷, YeaZ⁶⁸, the 2018 Kaggle Data Science Bowl dataset (DSB)⁶⁹, a collection of H&E datasets^{70–76}, and an internally collected dataset of phase microscopy images across eight mammalian cell lines (Phase400). We group these datasets into six types for evaluation: Tissue, Cell Culture, H&E, Bacteria, and Yeast. As the DSB⁶⁹ comprises cell nuclei that span several of these types, we evaluate it separately and refer to it as Nuclear, making a total of six categories for evaluation. While our method focuses on whole-cell segmentation, we included DSB⁶⁹ because cell nuclei are often used as a surrogate when the information necessary for whole-cell segmentation (e.g., cell membrane markers)

is absent from an image. Fig. 4.2a shows the number of annotations per evaluation type. Finally, we used a held-out dataset LIVECell⁶³ to evaluate CellSAM’s zero-shot performance. This dataset was curated to remove low-quality images and images that did not contain sufficient information about the boundaries of closely packed cells. A detailed description of data sources and preprocessing steps can be found in Methods 4.4. Our full, preprocessed dataset is publicly available at <https://cellsam.deepcell.org>.

CellSAM creates masks using box prompts generated from CellFinder

In early experiments, we found that providing ground truth bounding boxes as prompts to SAM (ground truth prompts represent an upper bound on performance) achieved significantly higher zero-shot performance than point prompting (see Fig. 4.4). This is in agreement with previous analysis of SAM applied to biological⁶¹ and medical⁵³ images. Because the ground-truth bounding box prompts yield accurate segmentation masks from SAM across various datasets, we sought to develop an object detector that could generate prompts for SAM in an automated fashion. Given that our zero-shot experiments demonstrated that ViT features can form robust internal representations of cellular images, we reasoned that we could build an object detector using the image features generated by SAM’s ViT. Previous work has explored this space and demonstrated that ViT backbones can achieve SOTA performance on natural images^{78,79}. For our object detection module, we use the Anchor DETR⁶² framework with the same ViT backbone as the SAM module; we call this object detection module CellFinder. Anchor DETR is well-suited for object detection in cellular images because it formulates object detection as a set prediction task. This allows it to perform cell segmentation in images with densely packed objects, a common occurrence in cellular imaging data. Alternative bounding box methods (e.g., the R-CNN family) rely on non-maximum suppression^{80,81}, leading to poor performance in this regime. Methods that frame cell segmentation as a dense, pixel-wise prediction task (e.g., Mesmer²⁸, Cellpose²⁵, and Hover-net³⁰) assume that each pixel can be uniquely assigned to a single cell and cannot handle overlapping objects.

The ground-truth prompting scheme by itself does not achieve real-world performance standards. Our analysis showed that SAM cannot accurately segment many cell types, likely due to the distribution of images seen during training. To adapt CellSAM from natural images to cellular images, we fine-tune the SAM model neck (the layers connecting SAM’s ViT to its decoder) while leaving other layers

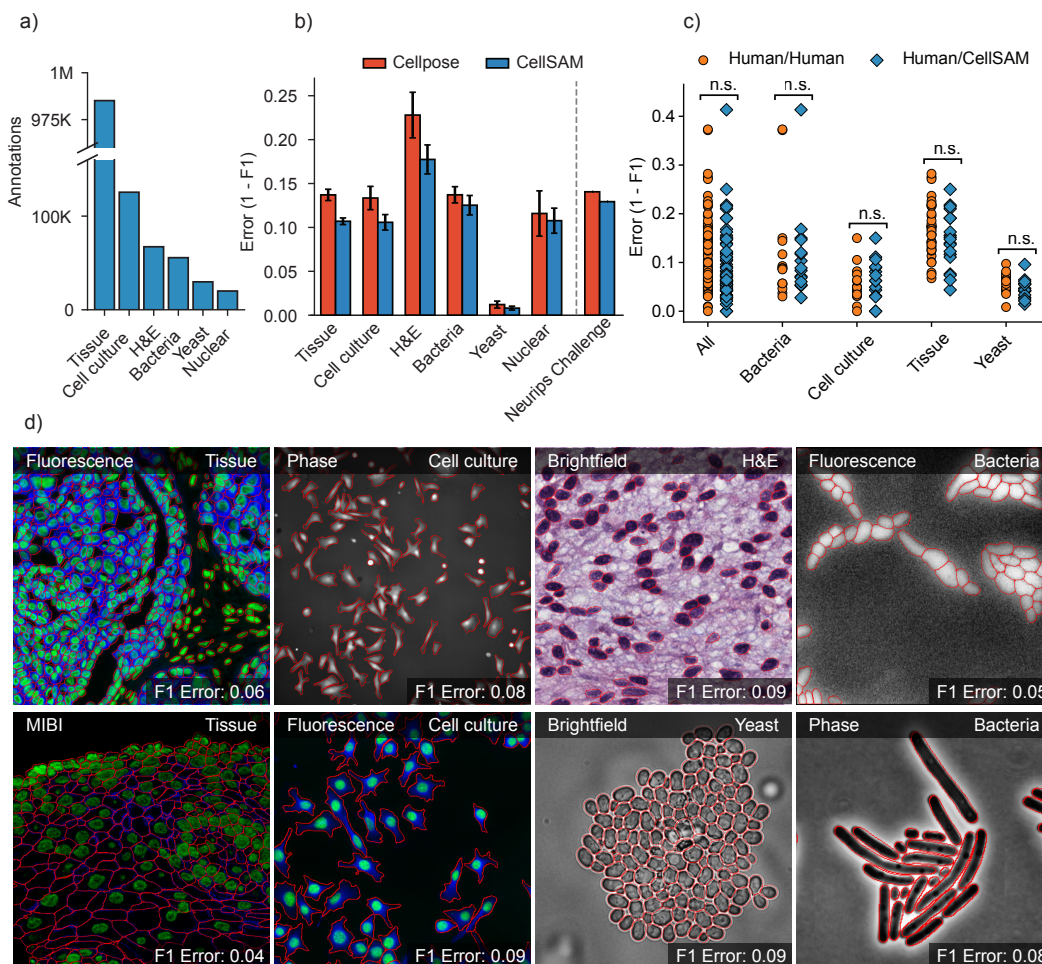


Figure 4.2: CellSAM is a strong generalist model for cell segmentation. a) For training and evaluating CellSAM, we curated a diverse cell segmentation dataset from the literature. The number of annotated cells is given for each data type. Nuclear refers to a heterogeneous dataset (DSB)⁶⁹ containing nuclear segmentation labels. b) Segmentation performance for CellSAM and Cellpose across different data types. We compared the segmentation error (1-F1) for models that were trained as generalists (i.e., the full dataset). Models were trained for a similar number of steps across all datasets. We observed that CellSAM-generalist had a lower error than Cellpose-generalist on all tested data categories. Furthermore, we validated this finding on a held-out competition dataset from the Weakly Supervised Cell Segmentation in Multi-modality High-Resolution Microscopy Images (i.e., the NeurIPS Challenge)⁷⁷. Error bars were computed by calculating the segmentation error per image, and then calculating the mean and standard error. The categories contained the following number of samples: Tissue = 330, Cell Culture = 144, H&E=51, Bacteria=260, Yeast=32, and Nuclear=56. c) Human vs human and CellSAM-generalist vs human (CS/human) inter-rater performance comparison. A two-sided t-test confirms that no statistical difference exists between CellSAM and human performance. d) Qualitative results of CellSAM segmentations for different data and imaging modalities. Predicted segmentations are outlined in red.

frozen to retain generalization ability. Training CellSAM in this manner, achieved state-of-the-art accuracy when provided with ground-truth bounding box prompts.

We train CellSAM in two stages; the full details can be found in the Methods. In the first stage, we train CellFinder on the object detection task. We convert the ground truth cell masks into bounding boxes and train the ViT backbone and the CellFinder module. Once CellFinder is trained, we freeze the model weights of the ViT and fine-tune the SAM module as described above. This accounts for the distribution shifts in the ViT features that occur during the CellFinder training. Once training is complete, we use CellFinder to prompt SAM’s mask decoder. We refer to the collective method as CellSAM; Fig. 4.1 outlines an image’s full path through CellSAM during inference.

Benchmarking CellSAM’s performance on numerous biological datasets

We benchmarked CellSAM’s performance using F1 error (1-F1) as metric (Fig. 4.2b) against Cellpose, a widely used cell segmentation algorithm. Because our work includes both dataset and model development, we chose benchmarks that allow us to measure the contributions of data and model architecture to overall performance. Our benchmarks include comparisons to a pre-trained generalist Cellpose model (cyto3), an internally trained generalist Cellpose model, and a suite of internally trained specialist (i.e., trained on a single dataset) Cellpose models. Internally trained models were trained on the CellSAM dataset or a suitable subset using previously published training recipes, while evaluations were performed on a held-out split of the same dataset. We further evaluated CellSAM’s performance on the evaluation split of the NeurIPS Cell Segmentation challenge⁸² (Fig. 4.2b). For this evaluation, we fine-tuned CellSAM with an additional hematology dataset, which was a substantial fraction of the NeurIPS challenge dataset. In almost every comparison, we found that CellSAM outperformed generalist Cellpose models (whether pre-trained or internally trained) and was equivalent to specialist Cellpose models trained exclusively on individual datasets. We highlight features of our benchmarking analyses below.

- **CellSAM is a strong generalist model.** Generalization across cell morphologies and imaging datasets has been a significant challenge for deep learning-based cell segmentation algorithms. To evaluate CellSAM’s generalization capabilities, we compared the performance of CellSAM and Cellpose models trained as specialists (i.e., on a single dataset) to generalists (i.e., on

all datasets). Consistent with the literature, we observe that Cellpose’s performance degraded when trained as a generalist (Fig. 4.6). In contrast, we found that the performance of CellSAM-generalist was equivalent or better than CellSAM-specialist across all data categories and datasets (Fig. 4.6). Moreover, CellSAM-generalist outperformed Cellpose-generalist in all data categories (Fig. 4.2b and Fig. 4.5). This analysis highlights an essential feature of a foundational model: maintaining performance with increasing data diversity and scale.

- CellSAM achieves human-level accuracy for generalized cell segmentation.** We use the error (1-F1) to assess the consistency of segmentation predictions and annotator masks across a series of images. We compared the annotations of three experts with each other (human vs. human) and with CellSAM (human vs. CellSAM). This comparison explores whether CellSAM’s performance is within the error margin created by annotator preferences (e.g., the thickness of a cell boundary). We compared annotations across four data categories: mammalian cells in tissue, mammalian cells in cell culture, bacterial cells, and yeast cells. A two-sided t-test revealed no significant differences between these two comparisons, indicating that CellSAM’s outputs are comparable to expert human annotators (Fig. 4.2c). This is demonstrated by non-significant p-values between CellSAM-annotator and inter-annotator agreements, specifically for Tissue: $P = 0.18$, Cell Culture: $P = 0.49$, Yeast: $P = 0.11$, Bacteria: $P = 0.90$.
- CellSAM enables fast and accurate labeling.** When provided with ground truth bounding boxes, CellSAM achieves high-quality cell masks without any fine-tuning on unseen datasets (Figure 4.7). Because drawing bounding boxes consumes significantly less time than drawing individual masks, this means CellSAM can be used to generate highly accurate labels quickly, even for out-of-distribution data.
- CellSAM is a strong zero-shot and few-shot learner.** We used the LIVE-Cell dataset to explore CellSAM’s performance in zero-shot and few-shot settings. We stratified CellSAM’s zero-shot by cell lines present in LIVE-Cell (Supplemental Fig. 4.7b). We found that while performance varied by cell line, we could recover adequate performance in the few-shot regime for a number of the cell lines (e.g., A172). Supplemental Figures 4.7 and 4.8 show that CellSAM improves its performance with only ten additional fields

of view ($10^2 - 10^3$ cells) for each cell line. We found fine-tuning could not recover performance for cell lines with morphologies far from the training data distribution (e.g., *SHSY5Y*). This may reflect a limitation of bounding boxes as a prompting strategy for SAM models.

CellSAM Enables Diverse Bioimage Analysis Workflows

Cell segmentation is a critical component of many spatial biology analysis pipelines; a single foundation model that generalizes across cell morphologies and imaging methods would fill a crucial gap in modern biological workflows by expanding the scope of the data that can be processed. In this section, we demonstrate how the same CellSAM-generalist model (not finetuned to any particular dataset) can be used across biological imaging pipelines by highlighting two use cases – spatial transcriptomics and live-cell imaging (Fig. 4.3).

Spatial transcriptomics methods measure single-cell gene expression while retaining the spatial organization of the sample. These experiments (e.g., MERFISH⁸³ and seqFISH⁸⁴) fluorescently label individual mRNA transcripts; the number of spots for a gene inside a cell corresponds to that gene’s expression level in that cell. These methods enable the investigation of spatial gene expression patterns from the sub-cellular to tissue scales but require accurate cell segmentation to yield meaningful insights. Here, we use CellSAM-generalist in combination with Polaris⁸⁵, a deep learning-enabled analysis pipeline for image-based spatial transcriptomics, to analyze gene expression at the single-cell level in MERFISH⁸⁶ and seqFISH⁸⁴ data (Fig. 4.3). With accurate segmentation, we can assign genes to specific cells (Fig. 4.3). We note that CellSAM can perform segmentation on either images of nuclear and membrane stains or images derived from the spots themselves (e.g., a maximum intensity projection of all spots). CellSAM’s ability to perform nuclear and whole-cell segmentation for challenging tissue images of dense cells with complex morphologies expands the scope of datasets to which Polaris can be applied.

The dynamics of cellular systems captured by live-cell imaging experiments elucidate various cellular processes such as cell division, morphological transitions, and signal transduction³⁴. The analysis of live-cell imaging data requires segmenting and tracking individual cells throughout whole movies. Here, we use CellSAM in combination with a cell tracking algorithm⁸⁷ (Fig. 4.3) in two settings. The first was a live cell imaging experiment with HeLa cells transiently expressing an AMP Kinase reporter⁸⁸ dosed with 20 mM 2-Deoxy-D-glucose, a setup reflective of many

experiments exploring cell signaling dynamics¹³. We imaged, segmented, and tracked the cells over 60 frames or 120 minutes to quantify AMP Kinase activity over time (Fig. 4.3). The second setting was lineage tracking in budding yeast cells. We again used CellSAM and cell tracking to segment and track cells; we further used a division detection algorithm to count the cumulative number of divisions over time and trace individual cell lineages (Fig. 4.3).

CellSAM can be used to segment 3D images. We developed a workflow that generates 3D segmentations by segmenting individual slices and aggregating them to 3D using the u-Segment3D⁸⁹ algorithm (Fig. 4.3b). We demonstrate on a thin slice of epidermal organoid⁸⁹, as well as EASI-FISH from the lateral hypothalamus⁹⁰.

CellSAM’s generality ensures consistent performance; in our experiments, we did not encounter catastrophic failure modes. This is in contrast to specialist models, where failures from inference on out-of-distribution data are common. CellSAM thus enables the analysis of data modalities for which specialist models do not exist (including any modality without widespread data). Furthermore, we simplify the analysis pipeline by eliminating the need for a large collection of potentially fragile specialist models. We note that use cases are representative of the modularity of bio-image analysis pipelines; most pipelines can be broken up into a few key steps. As CellSAM demonstrates, as the algorithms that perform these steps generalize, so does the entire pipeline.

4.3 Discussion

Cell segmentation is a critical task for cellular imaging experiments. While deep learning methods have made substantial progress in recent years, there remains a need for methods that can generalize across diverse images and further reduce the marginal cost of image labeling. In this work, we sought to meet these needs by developing CellSAM, a foundation model for cell segmentation. Transformer-based methods for cell segmentation have shown promising performance⁷⁷. CellSAM builds on these works by integrating the mask generation capabilities of SAM with transformer-based object detection to empower both scalable image labeling and automated inference. We trained CellSAM on a diverse dataset, and our benchmarking demonstrated that CellSAM achieves human-level performance on generalized cell segmentation. Compared to previous methods, CellSAM preserves its performance when trained on increasingly diverse data, which is essential for a foundational model. We found that CellSAM could be used on novel cell types

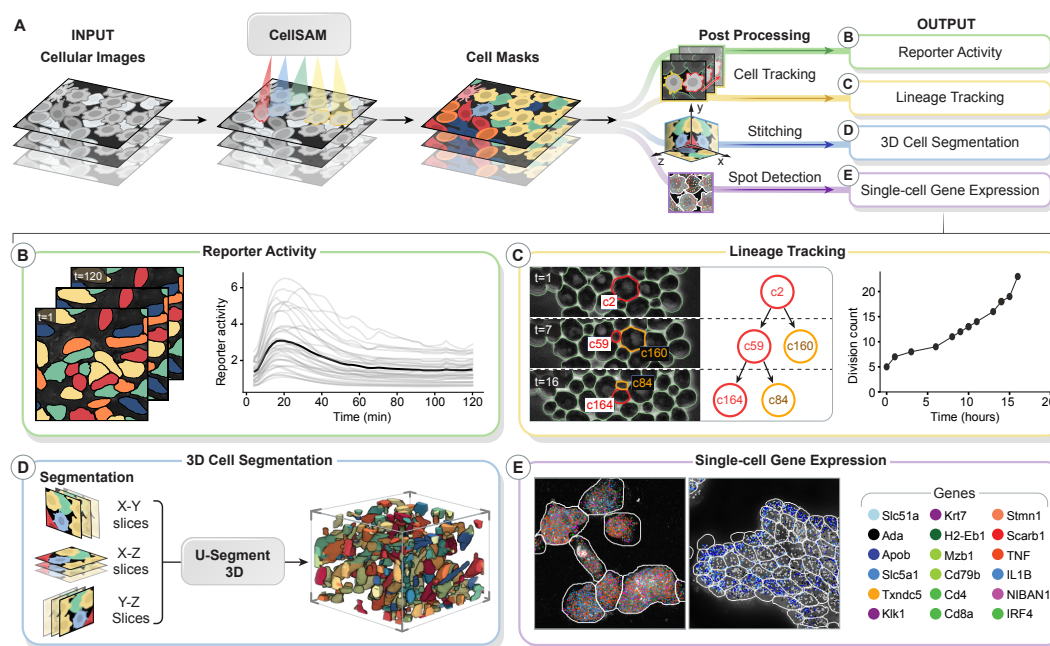


Figure 4.3: CellSAM Enables Diverse Bioimage Analysis Workflows. Because CellSAM-generalist functions across image modalities and cellular targets, it can be immediately applied across bioimaging analysis workflows without requiring task-specific adaptations. (a) We schematically depict how CellSAM-generalist fits into the analysis pipeline for live cell imaging and spatial transcriptomics, eliminating the need for different segmentation tools and expanding the scope of possible assays to which these tools can be applied. (b) Here, we demonstrate the application of CellSAM-generalist to various biological imaging tasks. (Top Left) Segmentations from CellSAM are used to track cells⁸⁷ and quantify fluorescent live-cell reporter activity in cell culture. (Top Right) CellSAM segments cells in multiple frames from a video of budding yeast cells. These cells are tracked across frames using a tracking algorithm⁸⁷ that ensures consistent identities, enabling accurate lineage construction and cell division quantification. (Bottom Left) CellSAM is used to segment slices of a 3D image, and these segmented slices are fed into a 3D U-Net⁸⁹ to create a 3D segmentation. (Bottom Right) Segmentations generated using CellSAM are integrated with Polaris⁸⁵, a spatial transcriptomics analysis pipeline. Because of CellSAM's generalist nature, we can apply this workflow across sample types (e.g., tissue and cell culture) and imaging modalities (e.g., seqFISH and MERFISH). Datasets of cultured macrophage cells (seqFISH) and mouse ileum tissue (MERFISH)⁸⁶ were used to generate the data in this example. MERFISH segmentations were generated with CellSAM with an image of a nuclear and membrane stain; seqFISH segmentations were generated with CellSAM with a maximum intensity projection image of all spots.

in a zero-shot setting, and that re-training with few labels could yield a strong boost in performance if needed. Moreover, we demonstrated that CellSAM’s ability to generalize can be extended to entire image analysis pipelines, as illustrated by use cases in spatial transcriptomics and live cell imaging. Given its utility in image labeling and high accuracy during inference, we believe CellSAM is a valuable contribution to the field, both as a tool for spatial biology and as a means to creating the data infrastructure required for cellular imaging’s AI-powered future. To facilitate the former, we have deployed a user interface for CellSAM at <https://cellsam.deepcell.org/> that allows for both automated and manual prompting. We also created a Napari plugin so that CellSAM can easily be integrated into existing workflows.

We note that for generalist models to achieve wide adoption, equivalent or superior performance compared to specialist models is necessary. CellSAM is, to our knowledge, the first generalist model where this has been demonstrated across a diverse array of imaging data. This marks a critical threshold, as it enables users to adopt a unified model without sacrificing task-specific performance.

The work described here is important beyond aiding life scientists with cell segmentation. First, foundation models are immensely useful for natural language and vision tasks and hold similar promise for the life sciences, provided they are suitably adapted to this new domain. We can see several uses for CellSAM that might be within reach of future work. Much like what has occurred with natural images, we foresee that the integration of natural language labels in addition to cell-level labels might lead to vision-language models capable of generating human-like descriptors of cellular images with entity-level resolution⁴⁸. More powerful generalization capabilities may enable the standardization of cellular image analysis pipelines across all the life sciences. If the accuracy is sufficient, microbiologists and tissue biologists could use the same collection of foundation models for interpreting their imaging data even for challenging experiments^{91,92}.

While the work presented here highlights the potential foundation models hold for cellular image analysis, much work remains to be done for this future to manifest. Extension of this methodology to 3D imaging data is essential; recent work on memory-efficient attention kernels⁹³ will aid these efforts. Exploring how to enable foundation models to leverage the full information content of images (e.g., multiple stains, temporal information for movies, etc.) is an essential avenue of future work. While CellSAM generalizes well across many different cell types, performance

guarantees can only be made for images similar to the training data. This was evidenced by our zero-shot evaluation on LIVECell. While foundation models hold much promise, care must be taken with claims surrounding generalization. The most reliable way of obtaining generalization is in building the support of the training data distribution; the human-in-the-loop capabilities supported by SAM-like models can aid this effort. Generally, expanding the space of labeled data remains a priority - this includes images of perturbed cells and cells with more challenging morphologies (e.g., neurons) where images contain many overlapping cells. Data generated by pooled optical screens⁹⁴ may synergize well with the data needs of foundation models. Compute-efficient fine-tuning strategies must be developed to enable flexible adaptation to new image domains. Lastly, prompt engineering is a critical area of future work, as it is essential to maximizing model performance. The work we present here can be thought of as prompt engineering, as we leverage CellFinder to produce bounding box prompts for SAM. As more challenging labeled datasets are incorporated, the nature of the “best” prompts will likely evolve. Finding the best prompts for these new data is a task that will likely fall on both the computer vision and life science communities.

4.4 Methods

Dataset Construction

To train CellSAM, we combined ten separate datasets spanning a variety of modalities: TissueNet²⁸, DeepBacs⁶⁴, BriFiSeg⁶⁵, Cellpose^{25,26}, Omnipose⁶⁶, YeastNet⁶⁷, YeaZ⁶⁸, the 2018 Kaggle Data Science Bowl (DSB)⁶⁹, a collection of H&E datasets^{70–76}, and an internally collected dataset of phase microscopy images across eight mammalian cell lines (Phase400). The LIVECell⁶³ dataset was held out for zero-shot/few-shot tests. Our collective dataset included images across multiple imaging modalities (brightfield, phase contrast, fluorescence, and mass cytometry), imaging targets (histology sections, yeast, cell culture, bacteria, nuclei), length scales, and morphologies. We did not do any preprocessing and left pixel intensities untouched. We treated nuclear and whole-cell channels as green and blue channels in an RGB image, respectively, and the red channel was always blank. We moved the green channel to blue for nuclear-only datasets (i.e., BriFiSeg and DSB) to keep the blue channel always occupied.

If available, we used predetermined training/validation/test splits for each dataset; otherwise, we introduced 80-10-10% data splits. For datasets with multiple fields of view of the same object set, we required all FOVs to belong to the same split. We deferred all duplicated samples to the train split for published datasets with a pre-existing data leak (detected by pixel-wise hashing). Our assembled dataset kept all images in their original size. We followed a widely used annotation scheme for labeling our masks, with zero representing the background and unique positive integers representing different objects. While this format precludes accurate segmentation of overlapping objects, labels of this kind were not present in the dataset we compiled. We filtered out invalid cell labels if the label contained disjoint regions or if the label had only a 1-pixel height or width. The processed images with filtered annotations were used for training, validation, and testing. We conducted some additional processing for LIVECell⁶³. We converted annotations from the COCO format to the same labeling format we used on the other datasets for consistency. We used Cellpose’s²⁶ pre-processing function `livecell_ann_to_masks()` to remove overlapping regions. In addition, we noticed inconsistencies in ground truth labels as previously observed by the Cellpose team (see Fig 1.c in²⁶). We thus manually inspected the LIVECell test split to divide the annotation quality into three classes - good, medium, and poor. We randomly selected images in the good split of the test set for the CellSAM few-shot learning task.

We modified our preparation pipeline for the NeurIPS challenge dataset. In addition to our standard preprocessing, we reduced bright spots by linearly re-scaling the raw pixel intensities such that the 99.9 percentiles corresponded to 1.0. We then normalized each image with Contrast Limited Adaptive Histogram Equalization (CLAHE)⁹⁵ with a kernel size of 128 pixels. Our assembled NeurIPS dataset used a fixed image size of 512 by 512 pixels. Images shorter than 512 pixels on either axis were zero-padded up to 512. For images with more than 512 pixels on either axis, we tiled them to 512 by 512 pixels with a 25% overlap and filled the empty regions with zeros. Any cropped images without valid annotations were removed. All images were upsampled to 1024x1024 as input for CellSAM. The NeurIPS training dataset includes all training/validation/test splits from our standard datasets, plus the NeurIPS training and tuning datasets. We used NeurIPS open test set for validation and the hidden test set for the performance report.

Statistics on our full dataset are in Table 4.1. To aid reproducibility, the dataset labels in Table 4.1 correspond to the labels present in our compiled dataset available at <https://cellsam.deepcell.org>.

CellSAM Architecture.

We adapted Anchor DETR⁶² for the object detector for CellSAM (CellFinder). This choice was motivated by Anchor DETR being non-maximum suppression (NMS)⁹⁶ free. NMS suppresses bounding boxes with a high amount of overlap to remove duplicate detections. While this works well for natural images, cellular images often have tightly clustered objects, and NMS-based methods such as the R-CNN family^{80,81} can suffer from a low recall in this setting. We replaced the Anchor DETR’s ResNet⁹⁷ backbone with the vision transformer (ViT)⁴⁴ from the SAM model⁵²; specifically, we used the base-sized ViT (ViT-B).

As the maximum number of cells per image is generally no more than 1000, we increased the number of queries q to 3500, 3.5 times the maximum number of cells, based on Fig. 12 in DETR⁹⁸, which provided an estimate of the number of queries needed for a DETR method to detect all objects. We used one pattern p for the Anchor generation as most objects in cellular detection are usually of similar scale.

Training CellFinder: We used a base learning rate of 10^{-4} for the Anchor DETR head and 10^{-5} for the SAM-ViT backbone. We used weight decay of 10^{-4} and clip norm of 0.1. We used AdamW⁹⁹ with a step-wise learning rate scheduler that drops the learning rate by a factor of 10 after the 1960th epoch. We trained CellFinder for

2800 epochs with a batch size of 4 across 8 H100 GPUs.

Finetuning CellSAM: After we trained CellFinder with the SAM-ViT backbone, the SAM-ViT output features were no longer aligned with the rest of the model (i.e., the prompt encoder and mask decoder). To close this distribution gap, we froze the SAM-ViT (such that it continues to function well with CellFinder) and trained the neck of the SAM model. We trained this neck using ground-truth bounding boxes as inputs and segmentation masks of individual cells as targets. We used a learning rate of 10^{-4} and weight decay of 10^{-4} for this training. We also used AdamW⁹⁹ for this training and did not clip the gradient.

We found this training recipe to be best with respect to computational cost and overfitting, based on small-scale ablation experiments.

Inference

At inference, we followed the following workflow. First, the input was passed through the Anchor DETR fine-tuned ViT-B. This resulted in an embedding dimension of 768. This embedding was then passed as an input to two parts of CellSAM, 1) the trained Anchor DETR module (CellFinder) and 2) the fine-tuned neck, which is a 2D convolutional network reducing the embedding dimensionality further to 256. The bounding box outputs of CellFinder were then sent into the prompt encoder, resulting in the prompt embedding. The prompt embeddings and neck embedding were then passed to the mask decoder, which outputs pixel-wise probabilities for the cell and another IoU-based confidence value for the prediction as a whole. This results in a tensor of shape $N \times W \times H$, where N corresponds to the number of cells predicted. This tensor was processed with a sigmoid and a threshold operation, resulting in binarized images. Depending on the metric used, we either used this tensor directly with the N scores (specifically for computation of the COCO AP @ 0.5 IoU), or we computed the argmax over the cell dimension N to generate a tensor $W \times H$, where each pixel corresponded to a unique integer label for each cell.

Thresholding: Given CellSAM’s model architecture, we had three different thresholds at inference time. First, we had a threshold on the bounding boxes generated by CellFinder, which we set to 0.4 across all datasets. We dynamically adjust this threshold by clustering the confidence values. We perform k-means clustering ($k = 2$) the box-confidences of all cells for a given image. We then compute the mean of both clusters (T_μ), i.e. the separation between likely non-cells and cells. We then adjust the final bounding box threshold as $T_{box} = \frac{2}{3} * 0.4 + \frac{1}{3} * T_\mu$. The

resulting boxes were passed through the Mask Decoder, we had an overall mask score outputted by the IoU prediction head of the Mask Decoder, which we set to 0.5. Lastly, we thresholded the mask decoder output after applying the sigmoid function to each pixel, which we set at 0.5.

CellSAM Postprocessing: We used the same postprocessing steps that are used by Cellpose³². This consisted of hole filling and island removal for each predicted cell.

Inference time: We compute the inference time in seconds as a function of the number of cells (Fig. 4.10a). To generate this plot, we randomly sampled two images from each dataset in our test set and filtered them to ensure a diverse range of cell counts. All benchmarks were performed on a single A6000 GPU. Unlike Cellpose, which requires only a single model pass, CellSAM performs one pass for each query (i.e., each cell). As a result, the wall-clock time scales (approximately) linearly with the number of cells. We also compare CPU and GPU inference times for both CellSAM and Cellpose on $n = 20$ randomly sampled images. Whereas for GPU based inference both CellSAM and Cellpose take less than 1 second per image, Cellpose takes on average almost 8 seconds per image on CPU and CellSAM close to 12 seconds on CPU.

Model Implementation and Training

CellSAM was implemented in PyTorch¹⁰⁰. For CellFinder we modified the official Anchor DETR repo¹. For CellSAM, we modified the official Segment Anything repo². We used PyTorch Lightning¹⁰¹ to scale the training. Prototyping was done using NVIDIA’s RTX 4090. We used machines with NVIDIA A6000s, A100s (40GB and 80GB versions), or H100s for the experiments in the paper. The overall training time for CellSAM-generalist was 6d 12h on 8× H100 GPUs for training CellFinder and an additional 1d for fine-tuning the complete CellSAM model (note fine-tuning was done on 1× 4090 GPU). For the specialist models, these numbers can be linearly scaled by the fraction of the total data used (e.g., for 1/9 of the time for 1/9 of the data). Below we have added a plot of the training curves for training CellFinder and CellSAM-generalist (Figure 4.11).

¹<https://github.com/megvii-research/AnchorDETR>

²<https://github.com/facebookresearch/segment-anything>

3D integration

We use u-segment3D⁸⁹ to generate 3D cellular segmentations. We first segment each 2D slice using the CellSAM-generalist model. We then remove small masks and fill holes using Cellpose’s postprocessing. We then use the u-segment3D⁸⁹ indirect method to fuse the 2D segmentations to 3D.

Benchmarking

We benchmarked the performance of CellSAM models against Cellpose^{25,26} trained on our compiled datasets.

Cellpose Model Training.

We utilized the command provided by the Cellpose developers to train our custom Cellpose models. Using Cellpose library version 3.0.11, we trained both specialist and generalist models from scratch²⁶.

```
python -m cellpose --train --train_size --use_gpu \
  --dir {} --test_dir {} \
  --img_filter _img --mask_filter _masks \
  --pretrained_model None --chan 3 --chan2 2
```

We kept all the hyper-parameters untouched. We used the SGD optimizer with a weight decay of 10^{-5} and a batch size of 8. We trained each model for 500 epochs with a base learning rate of 0.2. We used the default learning rate scheduler. The learning rate increased linearly from 0 to 0.2 over the first ten epochs, then decreased by a factor of 2 every 10 epochs after the 400th epoch. We trained each model on a single NVIDIA A6000 GPU. In total, we trained ten specialist models and one generalist model. When evaluating the performance of the Cellpose model, we followed the implementation of the cyto3 model evaluating pipeline. We first estimated the cell diameters per image using a trained size model. Then, we predicted the segmentation masks.

Metrics

We used the Metrics package in the Cellpose library^{25,26}. Predictions that match the ground truth labels (determined by a mask IoU ≥ 0.5) are true positives (TP), predictions with no matching ground truth labels are false positives (FP), and ground truth

labels without a valid match are false negatives (FN). For the human/human and CellSAM/human comparison we used a lower IoU threshold of 0.3 to increase the size of our statistic (reflecting higher inter-rater variability). The CellSAM/human comparisons were made against all three annotators. CellSAM’s predictions were compared to each annotator, and each annotator was compared to every other annotator for the human/human comparison. We computed the recall, precision, and F1 scores using the following formulas:

- Recall: $\text{recall} = \frac{TP}{TP+FN}$.
- Precision: $\text{precision} = \frac{TP}{TP+FP}$.
- F1: $F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$.

We also used COCO-style evaluation metrics⁷⁹ during CellFinder’s development. The COCO metrics are a widely used benchmark for assessing the object-level quality of object detection and instance segmentation methods. These metrics report Average Precision (AP), the area under the Precision-Recall curve for a given object class. In our case, we only had a single object class: cells. COCO-style AP is computed across IoU thresholds from 0.50 to 0.95 in increments of 0.05. We report the mean AP across all IoU thresholds, denoted as COCO mAP, as well as AP at IoU = 0.50, denoted as COCO AP₅₀, to quantify CellFinder’s performance. Because the object density is much higher in cellular images than in natural images, we modified the limit for the maximum number of detections from 100 to 10,000. We also fed the actual confidence score per binary prediction of the CellSAM model to the COCO evaluator. For the Cellpose models, we used a fixed confidence score of 1.0.

Hyperparameters

For training, there are two sets of parameters. The first is for training CellFinder, which we trained for 2800 epochs with a batch size of 4 (primarily constrained by GPU memory) across 8 GPUs. We used the AdamW optimizer with a learning rate of 10^{-5} for the backbone and 10^{-4} for the remainder of the model. We set weight decay 10^{-4} and dropout to 0.1. After 1960 epochs, we reduced the learning rate by a factor of 10. The full configuration file can be found in the GitHub repository.

The second set of parameters is fine-tuning CellSAM which we trained for 50 epochs with a cosine LR schedule. This final fine-tuning primarily adjusts smaller

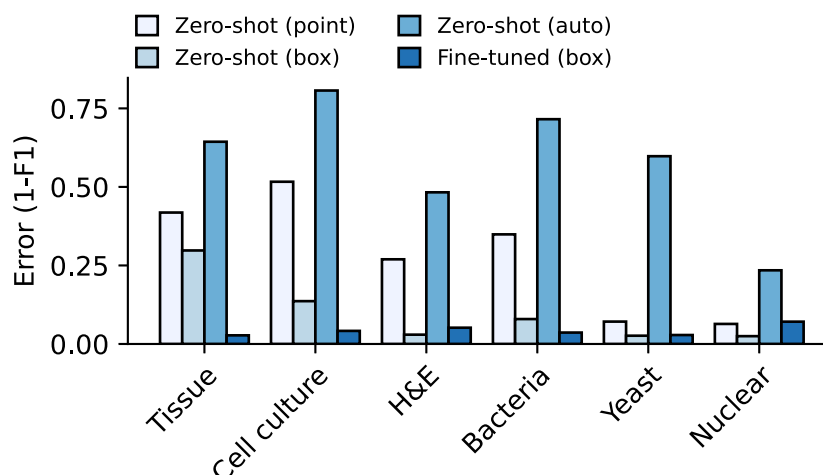


Figure 4.4: **Preliminary prompting analysis** comparing point prompts (zero-shot point), bounding box prompts (zero-shot box), and SAM’s auto prompting, a uniform grid of points, (zero-shot auto). We used the ground truth cellular masks to generate both the point and bounding box prompts (serving as a theoretical upper bound on performance). We used the F1 error (1-F1 score) for the metric of comparison. Additionally, we used SAM with no additional training to assess zero-shot performance and compared that to a fine-tuned SAM. Here, we see the best performance was achieved using bounding box prompts with a fine-tuned model. These bounding boxes can be generated using CellFinder in the full model. The categories contained the following number of samples (n): Tissue = 330, Cell Culture = 144, H&E = 51, Bacteria = 260, Yeast = 32, and Nuclear = 56.

components of CellSAM specifically, the "neck" (a feedforward neural network between the ViT and the rest of the model), which is prone to overfitting and thus requires careful tuning.

Data Availability

All datasets with test/training/validation splits are publicly available at <https://cellsam.deepcell.org>.

Code Availability

The code for CellSAM is publicly available at <https://github.com/vanvalenlab/cellsam>.

Extended Data

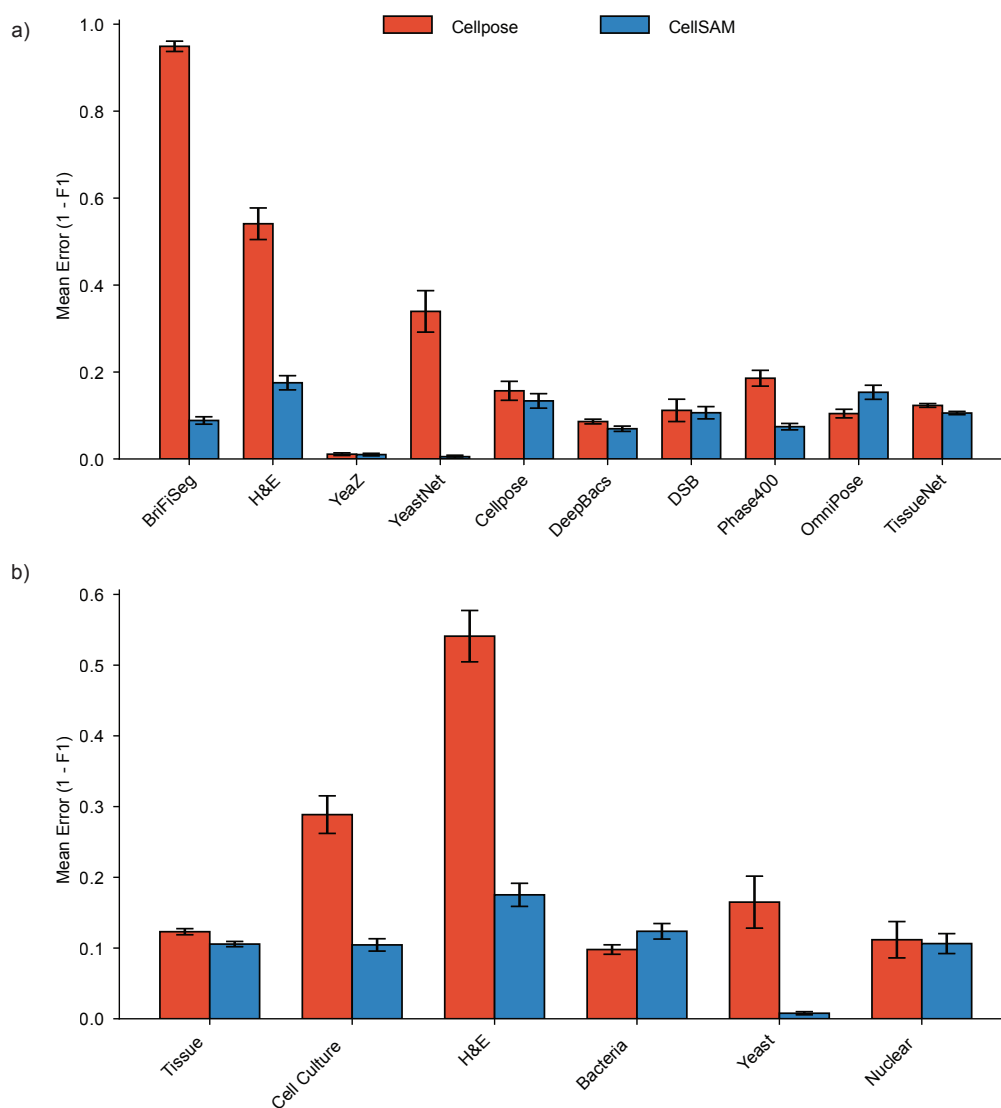


Figure 4.5: CellSAM-generalist Compared to Cellpose generalist model (cyto3).

Here, we compare CellSAM-generalist performance to Cellpose’s cyto3 model. a) examines the per dataset performance of these two models. The datasets contained the following number of samples: BriFiSeg = 22, H&E = 51, YeaZ = 17, YeastNet = 15, Cellpose = 68, DeepBacs = 92, DSB = 56, Phase400 = 54, OmniPose = 168, and TissueNet = 330. b) shows examples of the performance across the various data types. The categories contained the following number of samples: Tissue = 330, Cell Culture = 144, H&E = 51, Bacteria = 260, Yeast = 32, and Nuclear = 56.

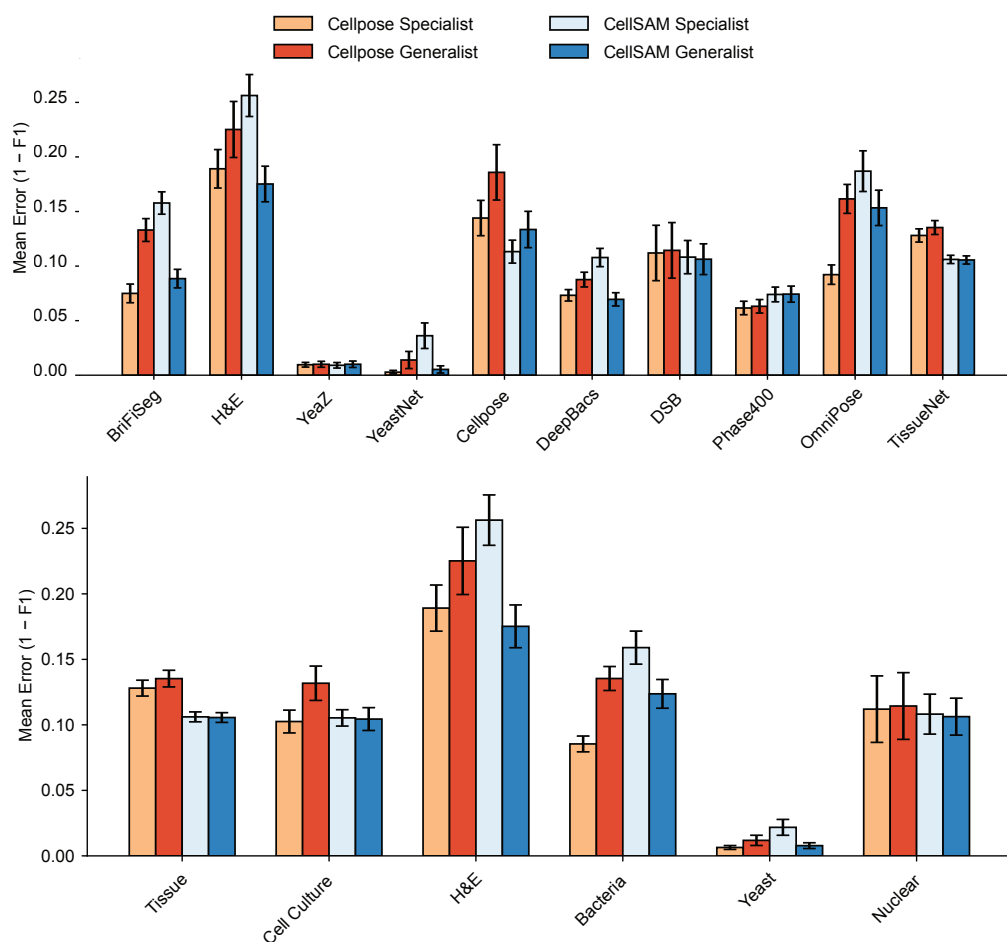


Figure 4.6: **Generalist vs. Specialist.** We compare the performance of CellSAM when trained on individual datasets (specialist) to when CellSAM is trained on all of the data (generalist). These results are compared to the performance of Cellpose when trained on individual datasets (specialist) and when Cellpose is trained on all the data (generalist). We see that CellSAM-generalist performs better across data types and datasets. Cellpose consistently performs better when trained on a specific dataset, but does not perform well when trained across multiple datasets.

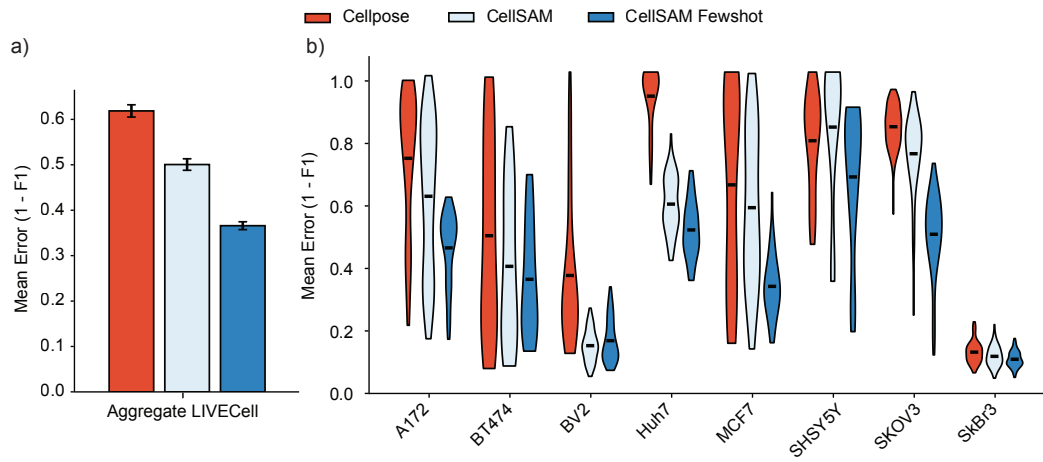
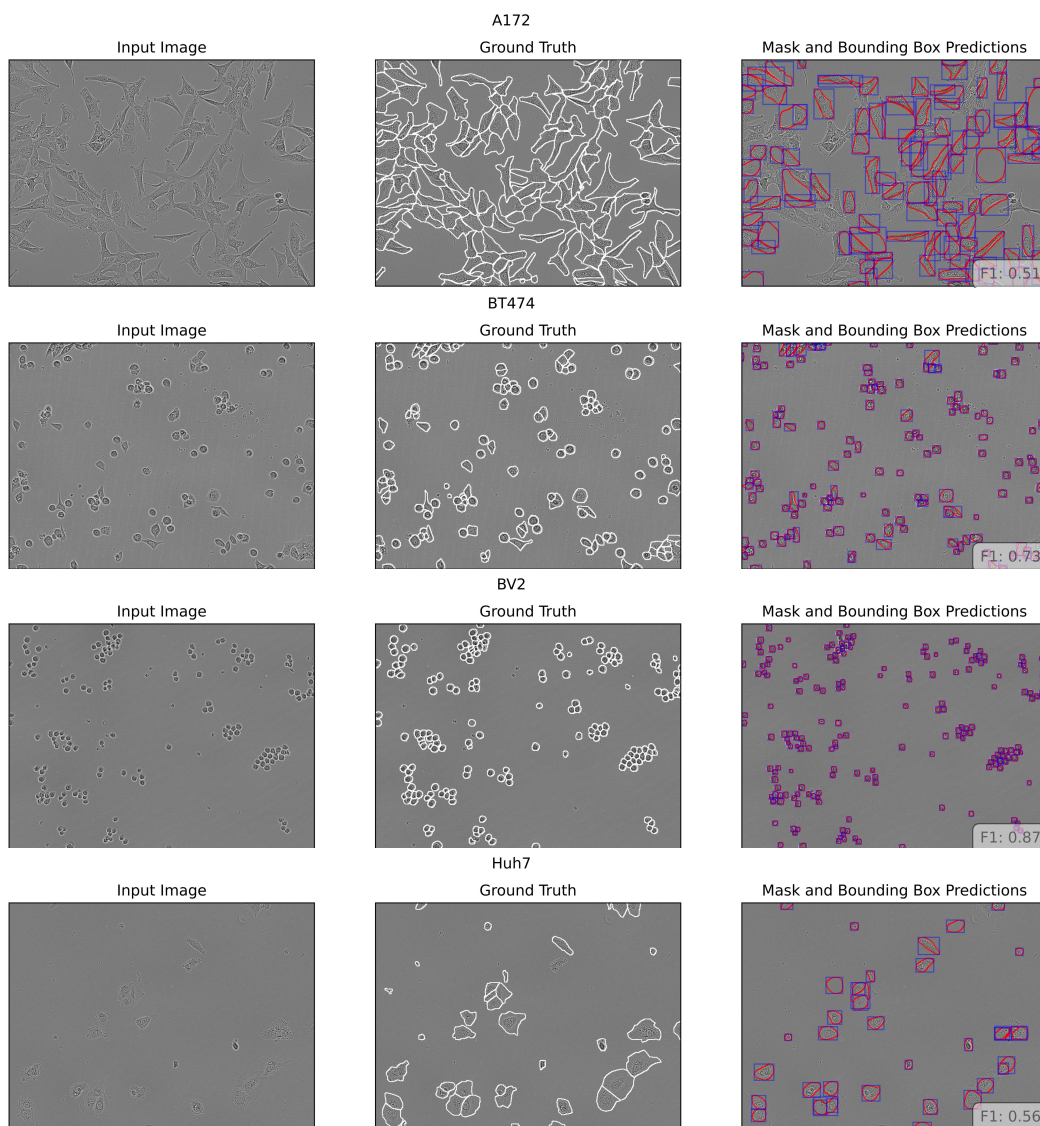


Figure 4.7: **LIVECell zero-shot and few-shot performance.** a) Zero-shot performance of CellSAM-generalist and Cellpose-generalist on the LIVECell dataset. Here, we show significant improvement of CellSAM over Cellpose on an unseen dataset (from 0.13 to 0.40 in F1). b) CellSAM-generalist performance stratified by cell line. We analyzed both zero-shot and few-shot (10 samples per cell line) performance. We saw that few-shot improves CellSAM-generalist on LIVECell for most cell lines.



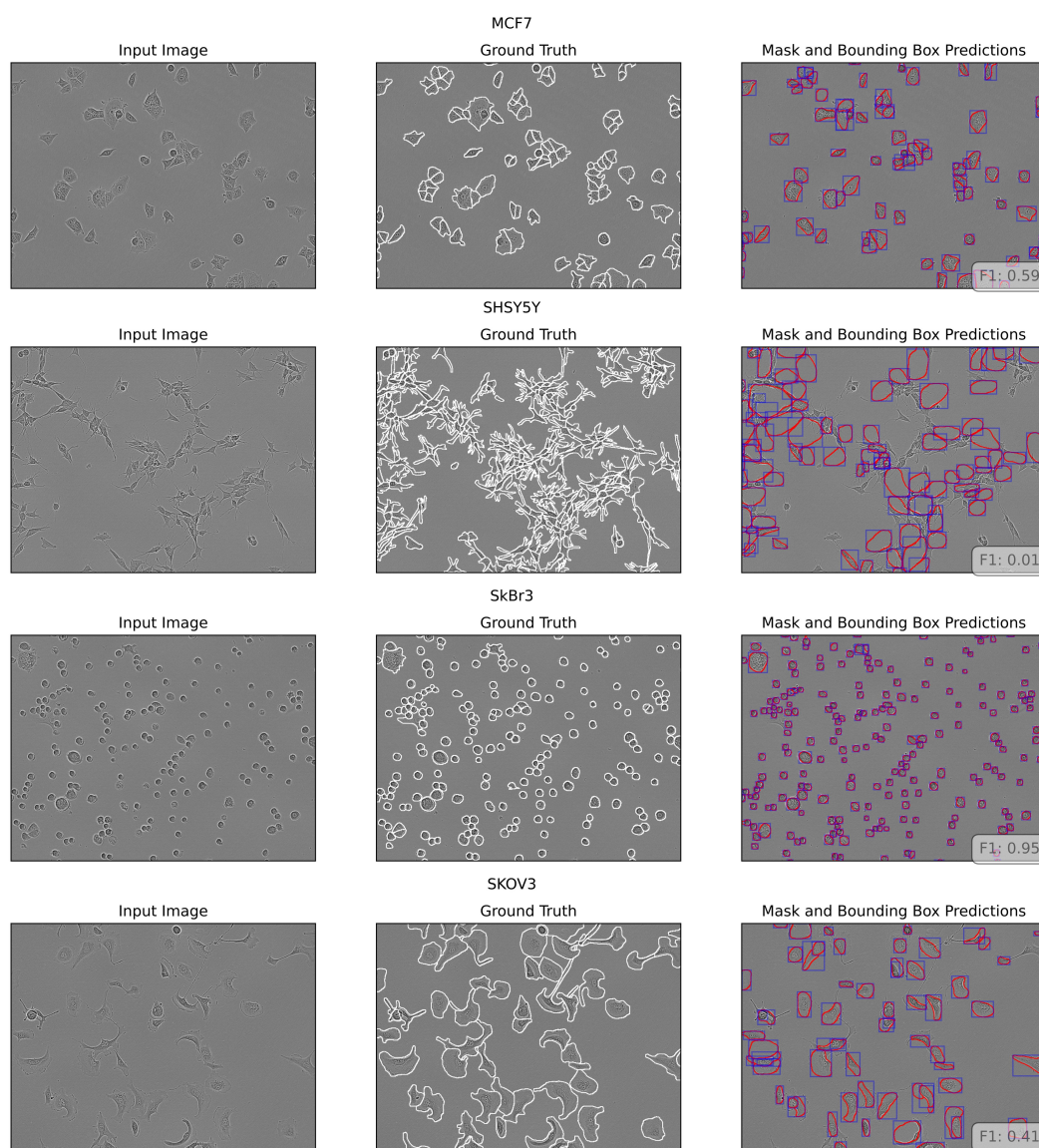


Figure 4.8: Representative qualitative results of CellSAM-generalist 10-shot performance on the LIVECell dataset, one panel for each cell line.

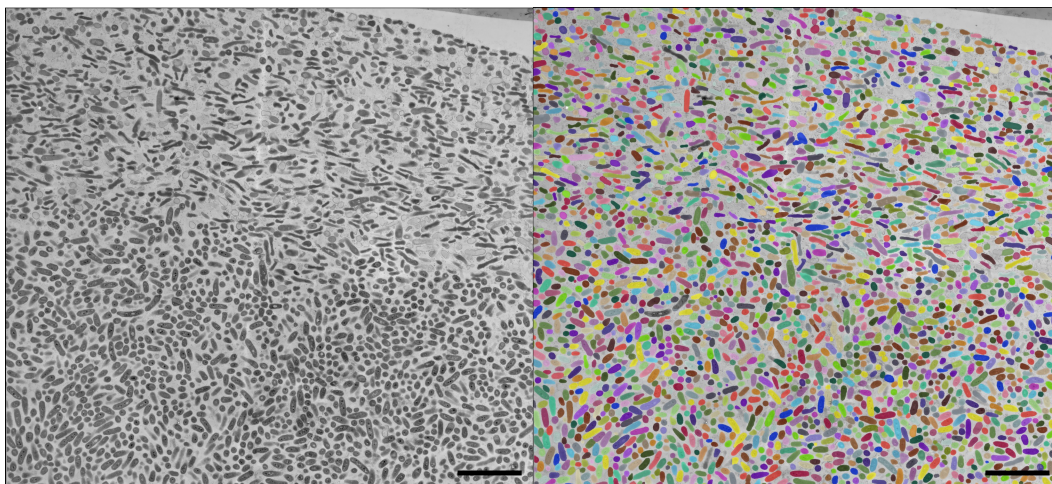


Figure 4.9: Zero-shot generalization of CellSAM-generalist to electron microscopy images of bacterial biofilms. The left panel shows the input electron microscopy image, and the right panel shows model predictions.

Supplementary Information

Dataset	Image size [height,width]	Pixel size (μm per pixel)	Test Images Objects	Train Images Objects	Val Images Objects
Tissue					
tissuenet_wholecell	[512,512]	0.365~1.0	330 132568	2576 985134	324 119915
Cell culture					
Cellpose	min:[164,174] max:[576,720] [512,512]	Various	68 7201	486 58701	54 5978
BriFiSeg		0.37	22 796	179 6458	23 815
Phase400	[1608, 1608]	0.55	4 658	32 4106	5 527
3T3		0.55	7 623	60 4950	8 483
A549		0.37	4 1002	29 6220	4 883
CHO		0.19	2 583	17 4051	3 659
HEK293		0.65	4 494	32 2789	4 353
HeLa		0.65	13 3034	103 25421	13 3368
HeLa-S3		0.55	7 696	54 4649	7 571
PC3		0.65	13 829	100 6968	13 1214
RAW264					
H&E					
cpm15	min:[392,459] max:[808,1032]	0.38	1 255	12 2048	2 602
cpm17	min:[500,500] max:[600,600]	0.31	16 1755	32 3766	16 2049
kumar	[1000,1000]	0.38	0 0	16 9905	0 0
monusac	min:[90,98] max:[1422,2162]	0.35	21 3864	167 23852	21 3482
monuseg	[1000,1000]	0.38	7 3822	37 24103	7 2861
nuinsseg	[512,512]	0.23	1 24	9 325	2 51
tnbc	[512,512]	0.40	5 391	40 3257	5 408
Bacteria					
Omnipose bact_fluor	min:[335,284] max:[2038,2038]	0.1 or 0.065	23 5388	129 16999	6 691
bact_phase	min:[66,58] max:[2038,2038]	0.1 or 0.065	145 18400	224 25567	22 1682
DeepBacs 2b_brightfield_dataset	[512,512]	0.14	5 470	2 394	3 425
2b_fluorescence_dataset	[512,512]	0.14	5 470	2 352	3 425
2c_e_coli	[1024,1024]	0.045	15 1142	4 335	4 252
2d_1_SplineDist_dataset	min:[512,512] max:[1024,1024]	0.025	10 649	22 385	10 138
2d_2_b_subtilis	[1024,1024]	0.065	10 649	70 3843	2 104
2e_e_coli	[512,512]	0.045	12 688	21 1083	4 269
s2_stardist	min:[256,256] max:[1024,1024]	0.55	35 2731	120 6656	8 476
Yeast					
YeastNet	[1024,1024]	0.1075	15 468	120 5988	15 894
YeaZ	min:[512,512] max:[2000,1988]	0.1075	17 1917	303 22526	8 852
Nuclear					
dsb_fixed	min:[256,256] max:[1040,1388]	Various	56 2292	449 20036	57 2558
NeurIPS challenge					
neurips_fixed	min:[200,200] max:[4096,4096]	Various	400 Not pub- licly avail- able	1099 213683	50 6041

Table 4.1: Statistics of datasets split by test, train, and val. For computing pixel size, we report values from the dataset source where available. In cases where the dataset does not report pixel size but the imaging camera is known, we report the pixel size of the camera. For other cases, we manually compute an estimation of the pixel size by taking the average major axis length over a subsample of 10 images and dividing by an estimator of the typical length scale for the celltype (e.g., 2 microns for a bacterial dataset). Note that this estimator can have high variance for diverse datasets, e.g., Cellpose, so we do not apply it.

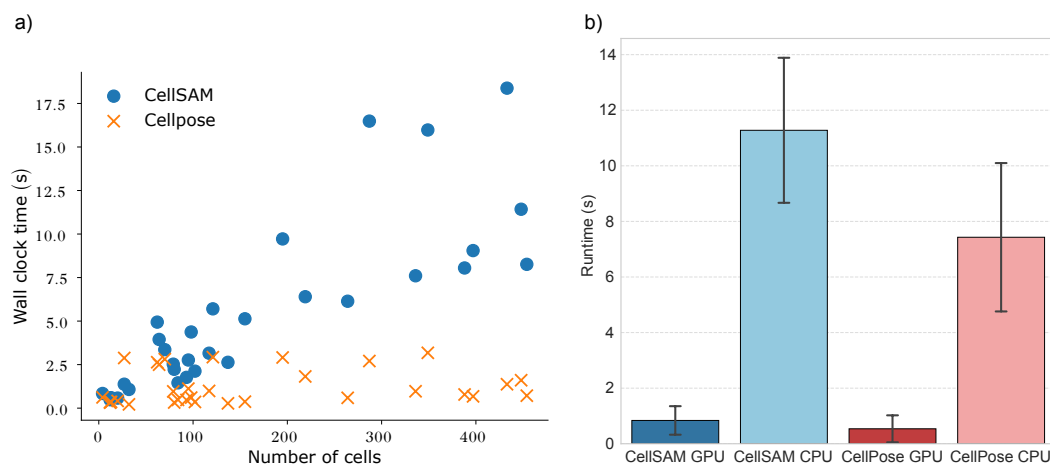


Figure 4.10: **Inference times for CellSAM and Cellpose.** a) Wall clock time over the number of cells. Cells were randomly sampled. b) GPU/CPU inference comparison for CellSAM and Cellpose. The Runtime in seconds is per image. N=20 were randomly sampled over multiple datasets. The Bar plot represents the mean and standard deviation.

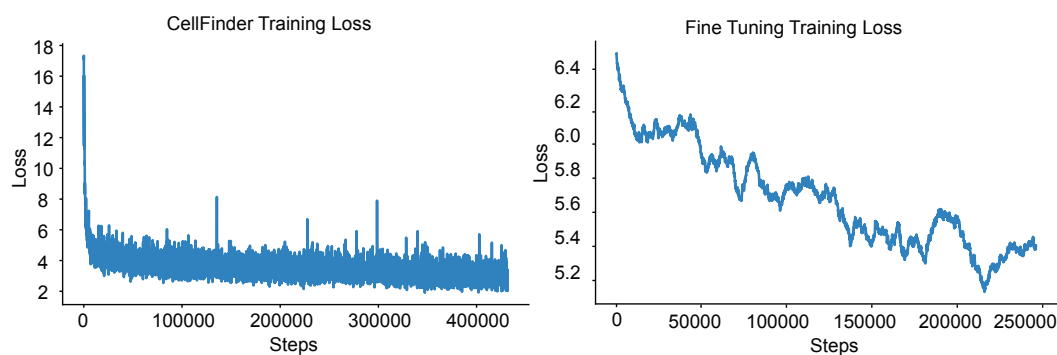


Figure 4.11: **Training Curves.** The training curves for training CellFinder (left) and fine-tuning the full CellSAM-generalist model (right) are shown here. We plot the training loss over training steps.

References

1. Giovanni Palla et al. “Spatial components of molecular tissue biology”. In: *Nature Biotechnology* 40.3 (Mar. 2022), pp. 308–318. doi: 10.1038/s41587-021-01182-1.
2. Jeffrey R. Moffitt, Emma Lundberg, and Holger Heyn. “The emerging landscape of spatial profiling technologies”. In: *Nature Reviews Genetics* 23.12 (Dec. 2022), pp. 741–759. doi: 10.1038/s41576-022-00515-3.
3. Lambda Moses and Lior Pachter. “Museum of spatial transcriptomics”. In: *Nature Methods* 19.5 (May 2022), pp. 534–546. doi: 10.1038/s41592-022-01409-2.
4. John W. Hickey et al. “Spatial mapping of protein composition and tissue organization: a primer for multiplexed antibody-based imaging”. In: *Nature Methods* 19.3 (Mar. 2022), pp. 284–295. doi: 10.1038/s41592-021-01316-y.
5. Madhav Mantri et al. “Spatiotemporal single-cell RNA sequencing of developing chicken hearts identifies interplay between cellular differentiation and morphogenesis”. In: *Nature Communications* 12.1 (Mar. 19, 2021). doi: 10.1038/s41467-021-21892-z.
6. Siyuan Wang et al. “Helical insertion of peptidoglycan produces chiral ordering of the bacterial cell wall”. In: *Proceedings of the National Academy of Sciences* 109.10 (Mar. 6, 2012). doi: 10.1073/pnas.1117132109.
7. Enrique Rojas, Julie A. Theriot, and Kerwyn Casey Huang. “Response of *Escherichia coli* growth rate to osmotic shock”. In: *Proceedings of the National Academy of Sciences* 111.21 (May 27, 2014), pp. 7807–7812. doi: 10.1073/pnas.1402591111.
8. Anders S Hansen and Erin K O’Shea. “Limits on information transduction through amplitude and frequency regulation of transcription factor activity”. In: *eLife* 4 (May 18, 2015). doi: 10.7554/eLife.06559.
9. Anders S Hansen and Erin K O’Shea. “Promoter decoding of transcription factor dynamics involves a trade-off between noise and control of gene expression”. In: *Molecular Systems Biology* 9.1 (Jan. 2013). doi: 10.1038/msb.2013.56.
10. Savaş Tay et al. “Single-cell NF- κ B dynamics reveal digital activation and analogue information processing”. In: *Nature* 466.7303 (July 8, 2010), pp. 267–271. doi: 10.1038/nature09145.
11. Sergi Regot et al. “High-Sensitivity Measurements of Multiple Kinase Activities in Live Single Cells”. In: *Cell* 157.7 (June 2014), pp. 1724–1734. doi: 10.1016/j.cell.2014.04.039.

12. Jangir Selimkhanov et al. “Accurate information transmission through dynamic biochemical signaling networks”. In: *Science* 346.6215 (Dec. 12, 2014), pp. 1370–1373. doi: 10.1126/science.1254933.
13. Jeremy E. Purvis et al. “p53 Dynamics Control Cell Fate”. In: *Science* 336.6087 (June 15, 2012), pp. 1440–1444. doi: 10.1126/science.1218351.
14. Maria Alieva et al. “Bridging live-cell imaging and next-generation cancer treatment”. In: *Nature Reviews Cancer* 23.11 (Nov. 2023), pp. 731–745. doi: 10.1038/s41568-023-00610-5.
15. Jianfeng Cao et al. “Establishment of morphological atlas of *Caenorhabditis elegans* embryo with cellular resolution using deep-learning-based 4D segmentation”. In: (Oct. 8, 2019). doi: 10.1101/797688.
16. Nicholas Moskwa et al. “Single-cell RNA sequencing reveals PDGFR α + stromal cell subpopulations that promote proacinar cell differentiation in embryonic salivary gland organoids”. In: *Development* 149.6 (Mar. 15, 2022). doi: 10.1242/dev.200167.
17. Kaitlin L. McDaniel et al. “Vitamin A-Deficient Hosts Become Nonsymptomatic Reservoirs of *Escherichia coli*-Like Enteric Infections”. In: *Infection and Immunity* 83.7 (July 2015), pp. 2984–2991. doi: 10.1128/iai.00201-15.
18. Jason M. Peters et al. “A Comprehensive, CRISPR-based Functional Analysis of Essential Genes in Bacteria”. In: *Cell* 165.6 (June 2016), pp. 1493–1506. doi: 10.1016/j.cell.2016.05.003.
19. Suckjoon Jun et al. “Fundamental principles in bacterial physiology—history, recent progress, and the future with focus on cell size control: a review”. In: *Reports on Progress in Physics* 81.5 (May 1, 2018), p. 056601. doi: 10.1088/1361-6633/aaa628.
20. Ping Wang et al. “Robust Growth of *Escherichia coli*”. In: *Current Biology* 20.12 (June 2010), pp. 1099–1103. doi: 10.1016/j.cub.2010.04.045.
21. Won-Ki Huh et al. “Global analysis of protein localization in budding yeast”. In: *Nature* 425.6959 (Oct. 2003), pp. 686–691. doi: 10.1038/nature02026.
22. Ellen C. O’Shaughnessy et al. “Tunable Signal Processing in Synthetic MAP Kinase Cascades”. In: *Cell* 144.1 (Jan. 2011), pp. 119–131. doi: 10.1016/j.cell.2010.12.014.
23. Nan Hao and Erin K O’Shea. “Signal-dependent dynamics of transcription factor translocation controls gene expression”. In: *Nature Structural & Molecular Biology* 19.1 (Jan. 2012), pp. 31–39. doi: 10.1038/nsmb.2192.
24. Morgan Schwartz et al. “Scaling biological discovery at the interface of deep learning and cellular imaging”. In: *Nature Methods* 20.7 (July 2023), pp. 956–957. doi: 10.1038/s41592-023-01931-x.

25. Carsen Stringer et al. “Cellpose: a generalist algorithm for cellular segmentation”. In: *Nature Methods* 18.1 (Jan. 2021), pp. 100–106. doi: 10.1038/s41592-020-01018-x.
26. Marius Pachitariu and Carsen Stringer. “Cellpose 2.0: how to train your own model”. In: *Nature Methods* 19.12 (Dec. 2022), pp. 1634–1641. doi: 10.1038/s41592-022-01663-4.
27. Uwe Schmidt et al. “Cell Detection with Star-Convex Polygons”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2018*. Springer, 2018, pp. 265–273. doi: 10.1007/978-3-030-00934-2_30. arXiv: 1806.03535.
28. Noah F. Greenwald et al. “Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning”. In: *Nature Biotechnology* 40.4 (Apr. 2022), pp. 555–565. doi: 10.1038/s41587-021-01094-0.
29. Reka Hollandi et al. “nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer”. In: *Cell Systems* 10.5 (May 2020), 453–458.e6. doi: 10.1016/j.cels.2020.04.003.
30. Simon Graham et al. “Hover-Net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images”. In: *Medical Image Analysis* 58 (Dec. 2019), p. 101563. doi: 10.1016/j.media.2019.101563.
31. Weikang Wang et al. “Learn to segment single cells with deep distance estimator and deep cell detector”. In: *Computers in Biology and Medicine* 108 (May 2019), pp. 133–141. doi: 10.1016/j.combiomed.2019.04.006.
32. Carsen Stringer and Marius Pachitariu. “Cellpose3: one-click image restoration for improved cellular segmentation”. In: *Nature Methods* 22.3 (Mar. 2025), pp. 592–599. doi: 10.1038/s41592-025-02595-5.
33. Fridtjof Lund-Johansen et al. “MetaMass, a tool for meta-analysis of subcellular proteomics data”. In: *Nature Methods* 13.10 (Oct. 2016), pp. 837–840. doi: 10.1038/nmeth.3967.
34. Morgan Sarah Schwartz et al. “Caliban: Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning”. In: (Oct. 13, 2019). doi: 10.1101/803205.
35. Ashish Vaswani et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008. URL: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>.
36. Rishi Bommasani et al. “On the opportunities and risks of foundation models”. In: *arXiv* (2021). arXiv: 2108.07258.

37. Tom B. Brown et al. “Language models are few-shot learners”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: <https://dl.acm.org/doi/abs/10.5555/3495724.3495883>.
38. OpenAI. “GPT-4 technical report”. In: *arXiv* (2023). arXiv: 2303.08774.
39. Zeming Lin et al. “Evolutionary-scale prediction of atomic-level protein structure with a language model”. In: *Science* 379.6637 (Mar. 17, 2023), pp. 1123–1130. doi: 10.1126/science.ade2574.
40. Nadav Brandes et al. “ProteinBERT: a universal deep-learning model of protein sequence and function”. In: *Bioinformatics* 38.8 (Apr. 12, 2022), pp. 2102–2110. doi: 10.1093/bioinformatics/btac020.
41. Ahmed Elnaggar et al. “ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.10 (Oct. 1, 2022), pp. 7112–7127. doi: 10.1109/tpami.2021.3095381.
42. Ali Madani et al. “ProGen: Language Modeling for Protein Generation”. In: (Mar. 8, 2020). doi: 10.1101/2020.03.07.982272.
43. Jacob Beal et al. “An End-to-End Workflow for Engineering of Biological Networks from High-Level Specifications”. In: *ACS Synthetic Biology* 1.8 (Aug. 17, 2012), pp. 317–331. doi: 10.1021/sb300030d.
44. Alexey Dosovitskiy et al. “An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations*. 2021. arXiv: 2010.11929.
45. Mathilde Caron et al. “Emerging Properties in Self-Supervised Vision Transformers”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021, pp. 9630–9640. doi: 10.1109/iccv48922.2021.00951.
46. Maxime Oquab et al. “DINOv2: learning robust visual features without supervision”. In: *Transactions on Machine Learning Research* (2024). URL: <https://openreview.net/forum?id=a68SUt6zFt>.
47. Yuxin Fang et al. “EVA: Exploring the Limits of Masked Visual Representation Learning at Scale”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2023, pp. 19358–19369. doi: 10.1109/cvpr52729.2023.01855.
48. Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763. URL: <https://proceedings.mlr.press/v139/radford21a.html>.

49. Jean-Baptiste Alayrac et al. “Flamingo: a visual language model for few-shot learning”. In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., 2022, pp. 23716–23736. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/960a172bc7fbf0177cccb411a7d800-Abstract-Conference.html.
50. Tom Henighan et al. “Scaling laws for autoregressive generative modeling”. In: *arXiv* (2020). arXiv: 2010.14701.
51. Xiaohua Zhai et al. “Scaling Vision Transformers”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022, pp. 1204–1213. doi: 10.1109/cvpr52688.2022.01179.
52. Alexander Kirillov et al. “Segment Anything”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4015–4026. arXiv: 2304.02643.
53. Maciej A. Mazurowski et al. “Segment anything model for medical image analysis: An experimental study”. In: *Medical Image Analysis* 89 (Oct. 2023), p. 102918. doi: 10.1016/j.media.2023.102918.
54. Yizhe Zhang et al. “Input Augmentation with SAM: Boosting Medical Image Segmentation with Segmentation Foundation Model”. In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2023, pp. 129–139. doi: 10.1007/978-3-031-47401-9_13.
55. Wenhui Lei et al. “MedLSAM: Localize and segment anything model for 3D CT images”. In: *Medical Image Analysis* 99 (Jan. 2025), p. 103370. doi: 10.1016/j.media.2024.103370.
56. Jun Ma et al. “Segment anything in medical images”. In: *Nature Communications* 15.1 (Jan. 22, 2024). doi: 10.1038/s41467-024-44824-z.
57. Ting Chen et al. “A Generalist Framework for Panoptic Segmentation of Images and Videos”. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 1, 2023, pp. 909–919. doi: 10.1109/iccv51070.2023.00090.
58. Mingzhe Hu, Yuheng Li, and Xiaofeng Yang. “SkinSAM: empowering skin cancer segmentation with Segment Anything Model”. In: *arXiv* (2023). arXiv: 2304.13973.
59. Ruining Deng et al. “Segment Anything Model (SAM) for Digital Pathology: Assess Zero-shot Segmentation on Whole Slide Imaging”. In: *Electronic Imaging* 37.14 (Feb. 2, 2025), pp. 132-1-132-4. doi: 10.2352/ei.2025.37.14.coimg-132.
60. Fabian Hörst et al. “CellViT: Vision Transformers for precise cell segmentation and classification”. In: *Medical Image Analysis* 94 (May 2024), p. 103143. doi: 10.1016/j.media.2024.103143.

61. Anwai Archit et al. “Segment Anything for Microscopy”. In: *Nature Methods* 22.3 (Mar. 2025), pp. 579–591. DOI: 10.1038/s41592-024-02580-4.
62. Yingming Wang et al. “Anchor DETR: Query Design for Transformer-Based Detector”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.3 (June 28, 2022), pp. 2567–2575. DOI: 10.1609/aaai.v36i3.20158.
63. Christoffer Edlund et al. “LIVECell—A large-scale dataset for label-free live cell segmentation”. In: *Nature Methods* 18.9 (Sept. 2021), pp. 1038–1045. DOI: 10.1038/s41592-021-01249-6.
64. Christoph Spahn et al. “DeepBacs for multi-task bacterial image analysis using open-source deep learning approaches”. In: *Communications Biology* 5.1 (July 9, 2022). DOI: 10.1038/s42003-022-03634-z.
65. Mathieu Gendarme, Anne-Marie Lambert, and Bouchra El Debs. “BriFiSeg: a deep learning-based method for semantic and instance segmentation of nuclei in brightfield images”. In: *arXiv* (2022). arXiv: 2211.03072.
66. Kevin J. Cutler et al. “Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation”. In: *Nature Methods* 19.11 (Nov. 2022), pp. 1438–1448. DOI: 10.1038/s41592-022-01639-4.
67. Danny Salem et al. “YeastNet: Deep-Learning-Enabled Accurate Segmentation of Budding Yeast Cells in Bright-Field Microscopy”. In: *Applied Sciences* 11.6 (Mar. 17, 2021), p. 2692. DOI: 10.3390/app11062692.
68. Nicola Dietler et al. “A convolutional neural network segments yeast microscopy images with high accuracy”. In: *Nature Communications* 11.1 (Nov. 12, 2020). DOI: 10.1038/s41467-020-19557-4.
69. Juan C. Caicedo et al. “Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl”. In: *Nature Methods* 16.12 (Dec. 2019), pp. 1247–1253. DOI: 10.1038/s41592-019-0612-7.
70. Neeraj Kumar et al. “A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology”. In: *IEEE Transactions on Medical Imaging* 36.7 (July 2017), pp. 1550–1560. DOI: 10.1109/tmi.2017.2677499.
71. Amirreza Mahbod et al. “CryoNuSeg: A dataset for nuclei instance segmentation of cryosectioned H&E-stained histological images”. In: *Computers in Biology and Medicine* 132 (May 2021), p. 104349. DOI: 10.1016/j.compbimed.2021.104349.
72. Amirreza Mahbod et al. “NuInsSeg: A fully annotated dataset for nuclei instance segmentation in H&E-stained histological images”. In: *Scientific Data* 11.1 (Mar. 14, 2024). DOI: 10.1038/s41597-024-03117-2.
73. Peter Naylor et al. “Segmentation of Nuclei in Histopathology Images by Deep Regression of the Distance Map”. In: *IEEE Transactions on Medical Imaging* 38.2 (Feb. 2019), pp. 448–459. DOI: 10.1109/tmi.2018.2865709.

74. Neeraj Kumar et al. “A Multi-Organ Nucleus Segmentation Challenge”. In: *IEEE Transactions on Medical Imaging* 39.5 (May 2020), pp. 1380–1391. doi: 10.1109/tmi.2019.2947628.
75. Quoc Dang Vu et al. “Methods for Segmentation and Classification of Digital Microscopy Tissue Images”. In: *Frontiers in Bioengineering and Biotechnology* 7 (Apr. 2, 2019). doi: 10.3389/fbioe.2019.00053.
76. Ruchika Verma et al. “MoNuSAC2020: A Multi-Organ Nuclei Segmentation and Classification Challenge”. In: *IEEE Transactions on Medical Imaging* 40.12 (Dec. 2021), pp. 3413–3423. doi: 10.1109/tmi.2021.3085712.
77. NeurIPS Challenge Organizers. *NeurIPS 2022 Cell Segmentation Challenge*. 2022. URL: <https://neurips22-cellseg.grand-challenge.org/>.
78. Yanghao Li et al. “Exploring Plain Vision Transformer Backbones for Object Detection”. In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 280–296. doi: 10.1007/978-3-031-20077-9_17.
79. Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Springer, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48. arXiv: 1405.0312.
80. Ross Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2014, pp. 580–587. doi: 10.1109/cvpr.2014.81.
81. S. Ren et al. “Faster R-CNN: towards real-time object detection with region proposal networks”. In: *Advances in Neural Information Processing Systems*. 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.
82. Jun Ma et al. “The multimodality cell segmentation challenge: toward universal solutions”. In: *Nature Methods* 21.6 (June 2024), pp. 1103–1113. doi: 10.1038/s41592-024-02233-6.
83. Kok Hao Chen et al. “Spatially resolved, highly multiplexed RNA profiling in single cells”. In: *Science* 348.6233 (Apr. 24, 2015). doi: 10.1126/science.aaa6090.
84. Chee-Huat Linus Eng et al. “Transcriptome-scale super-resolved imaging in tissues by RNA seqFISH+”. In: *Nature* 568.7751 (Apr. 2019), pp. 235–239. doi: 10.1038/s41586-019-1049-y.
85. Emily Laubscher et al. “Accurate single-molecule spot detection for image-based spatial transcriptomics with weakly supervised deep learning”. In: *Cell Systems* 15.5 (May 2024), 475–482.e6. doi: 10.1016/j.cels.2024.04.006.

86. Viktor Petukhov et al. “Cell segmentation in imaging-based spatial transcriptomics”. In: *Nature Biotechnology* 40.3 (Mar. 2022), pp. 345–354. DOI: 10.1038/s41587-021-01044-w.
87. Erik Bochinski, Volker Eiselein, and Thomas Sikora. “High-Speed tracking-by-detection without using image information”. In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, Aug. 2017, pp. 1–6. DOI: 10.1109/avss.2017.8078516.
88. Danielle L. Schmitt et al. “Spatial regulation of AMPK signaling revealed by a sensitive kinase activity reporter”. In: *Nature Communications* 13.1 (July 5, 2022). DOI: 10.1038/s41467-022-31190-x.
89. Fang-Yi Zhou et al. “Universal consensus 3D segmentation of cells from 2D segmented stacks”. In: *bioRxiv* (2024). DOI: 10.1101/2024.05.03.592249.
90. Yuhan Wang et al. “EASI-FISH for thick tissue defines lateral hypothalamus spatio-molecular organization”. In: *Cell* 184.26 (Dec. 2021), 6361–6377.e24. DOI: 10.1016/j.cell.2021.11.024.
91. Sheel Shah et al. “In Situ Transcription Profiling of Single Cells Reveals Spatial Organization of Cells in the Mouse Hippocampus”. In: *Neuron* 92.2 (Oct. 2016), pp. 342–357. DOI: 10.1016/j.neuron.2016.10.001.
92. Daniel Dar et al. “Spatial transcriptomics of planktonic and sessile bacterial populations at single-cell resolution”. In: *Science* 373.6556 (Aug. 13, 2021). DOI: 10.1126/science.abi4882.
93. Eric Nguyen et al. “HyenaDNA: long-range genomic sequence modeling at single nucleotide resolution”. In: *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., 2023, pp. 43177–43201. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/86ab6927ee4ae9bde4247793c46797c7-Abstract-Conference.html.
94. David Feldman et al. “Optical Pooled Screens in Human Cells”. In: *Cell* 179.3 (Oct. 2019), 787–799.e17. DOI: 10.1016/j.cell.2019.09.016.
95. Stephen M. Pizer et al. “Adaptive histogram equalization and its variations”. In: *Computer Vision, Graphics, and Image Processing* 39.3 (Sept. 1987), pp. 355–368. DOI: 10.1016/s0734-189x(87)80186-x.
96. Jan Hosang, Rodrigo Benenson, and Bernt Schiele. “Learning Non-maximum Suppression”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017, pp. 6469–6477. DOI: 10.1109/cvpr.2017.685.

97. Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778. DOI: 10.1109/cvpr.2016.90. arXiv: 1512.03385.
98. Nicolas Carion et al. “End-to-End Object Detection with Transformers”. In: *Computer Vision – ECCV 2020*. Springer, 2020, pp. 213–229. DOI: 10.1007/978-3-030-58452-8_13. arXiv: 2005.12872.
99. Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *International Conference on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=Bkg6RiCqY7>.
100. Adam Paszke et al. “PyTorch: an imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html>.
101. William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. 2019. URL: <https://github.com/Lightning-AI/pytorch-lightning>.

FTURADER: FUNCTIONAL TISSUE UNIT SEGMENTATION WITH SCALE-DYNAMIC ADAPTATION

5.1 Introduction

Biological tissue organization is neither purely cellular nor purely organ-level. Between individual cells and gross organ anatomy are repeated multicellular structures that carry out organ-specific functions. These functional tissue units (FTUs) provide an intermediate scale for spatial biology: glomeruli filter blood in the kidney, intestinal crypts organize epithelial renewal, alveoli support gas exchange in the lung, glandular acini support secretion, and splenic white pulp coordinates immune function. FTUs therefore connect cell identity, local tissue architecture, and organ physiology in a way that single-cell masks alone cannot capture. Chapter 4 addressed segmentation of individual cells across imaging modalities; this chapter moves one level upward to ask which cells, extracellular regions, and luminal spaces belong to the same anatomical and functional unit.

This transition changes the segmentation problem. Cells are small, repeated, and often locally defined by membrane, nuclear, or cytoplasmic contrast. FTUs can span hundreds to thousands of pixels, contain empty luminal or air spaces, and require tissue context that extends beyond the immediate boundary. A white-pulp boundary in the spleen may depend on subtle transitions between immune-cell-rich and red-pulp regions; an alveolus may be missed when a tile contains mostly empty air space; and a glandular acinus may be confused with a larger glandular structure when the model lacks instance-level anatomical context. These properties make FTU segmentation more than a direct rescaling of cell segmentation.

Existing FTU benchmarks have shown the promise and limitations of community-developed semantic segmentation models. HuBMAP and Human Protein Atlas efforts have assembled multi-organ FTU segmentation data and motivated algorithms for glomeruli, intestinal crypts, alveoli, glandular acini, and splenic white pulp¹⁻³. These studies establish FTUs as practical objects for atlas construction, but they also expose a central gap: semantic foreground masks can be sufficient for pixel-overlap scoring, yet they do not directly provide the individual FTU objects needed for morphometry, per-unit cell composition, spatial indexing, or downstream

atlas registration. The FTU challenge paper notes that Dice coefficient does not account for separation between individual instances and that instance-level metrics such as mean average precision may be better suited for this object-aware task³.

A method can obtain a high pixel Dice score by covering the correct foreground area while merging neighboring FTUs, splitting one FTU into several fragments, or duplicating predictions over the same anatomical unit. Such outputs may appear acceptable under semantic overlap metrics but fail analyses that require FTU count, size, shape, and per-unit cellular composition. This chapter therefore treats FTU segmentation as an instance-segmentation problem and reports object-level metrics alongside pixel-level Dice and IoU.

FTURader is a detector-prompted FTU segmentation pipeline evaluated across five organ-associated FTU categories. A first-stage bounding-box detector, referred to here as FTUFinder, localizes candidate tissue units and estimates object scale. A dynamic-scaling module rescales images so that FTUs from different organs are presented to the mask decoder within a more stable apparent size range and partitions large inputs into manageable 1024×1024 -pixel tiles with 50% overlap. A SAM2-based promptable mask decoder then uses the detector-derived box and local tissue context to generate an FTU mask. This design follows the detector-plus-foundation-model pattern used for dense cell segmentation in CellSAM⁴, but adapts it to FTU-specific challenges: organ-dependent scale, stain variation, hollow structures, weak anatomical boundaries, and whole-slide tiling.

5.2 Results

FTURader converts histology images into FTU instances

FTURader takes an H&E histology tile or whole-slide-derived image region as input and returns a set of candidate FTU instances. Each output instance contains an FTU category, a confidence score, a bounding box, and a mask. The pipeline is organized into three modules. First, preprocessing reduces stain and intensity variation and filters empty image regions. Second, FTUFinder detects candidate FTUs and estimates their approximate object scale. Third, a SAM2-based decoder converts bounding-box prompts into instance masks. This modular design separates the questions of where an FTU is, at what scale it should be segmented, and how its segmentation mask should be decoded.

Figure 5.1 summarizes the pipeline. Images are first preprocessed to reduce stain and intensity variation and to filter uninformative regions. FTUFinder then predicts can-

didate FTU boxes, confidence scores, and scale-relevant localization information. The dynamic scaler rescales the image and, when needed, partitions large inputs into 1024×1024 -pixel tiles with 50% overlap. Finally, a SAM2-based promptable mask decoder uses the detector-derived box and local image features to generate one candidate FTU mask per prompt.

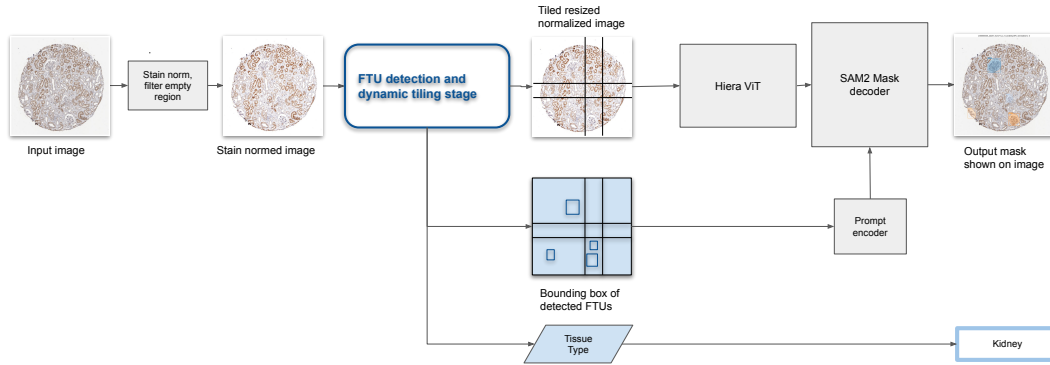


Figure 5.1: FTURader architecture for detector-prompted FTU segmentation. A histology tile is stain-normalized and filtered before candidate FTUs are localized by FTUFinder. FTUFinder supplies candidate boxes and confidence scores used by the downstream scale-and-tilde step. Candidate boxes are passed as prompts to a SAM2-based mask decoder, while the normalized image is tiled and encoded by a Hiera vision transformer. The decoder combines image features and box prompts to generate instance-level FTU masks.

Together, these components establish FTURader as a detector-guided framework for scale-normalized FTU instance segmentation. By combining object localization with promptable mask decoding, the pipeline converts histology images into candidate FTU instances that can support object-level morphometry, per-unit cellular analysis, and downstream tissue-atlas integration after appropriate expert validation.

Semantic FTU annotations are converted into candidate instance labels

The evaluation dataset is based on the HuBMAP and HPA “Hacking the Human Body” FTU segmentation setting, covering five organ-associated FTU categories: kidney glomeruli, large-intestine crypts, lung alveoli, prostate glandular acini, and splenic white pulp³. The source data provide semantic masks, which mark FTU foreground pixels without necessarily preserving individual object identity. For detector-prompted segmentation, these annotations were converted into candidate instance

masks and bounding boxes using the dataset providers’ official mask-transcoding workflow.

Table 5.1 summarizes the official split and organ-level annotation counts used in this chapter. The public training split contains 351 images and 4,894 candidate FTU annotations. The official public-test split was used internally for validation and threshold selection and contains 290 images and 4,414 annotations. The private test split contains 239 images and 3,565 annotations.

Table 5.1: FTU dataset split statistics. The public training split was used for model training, the official public-test split was used internally for validation and threshold selection, and the private test split was reserved for final held-out testing. “Annot.” reports candidate FTU instance annotations produced from the official mask-transcoding workflow.

Organ	Train		Validation		Test	
	Images	Annot.	Images	Annot.	Images	Annot.
Kidney	99	337	56	321	42	286
Large intestine	58	3,105	37	1,794	24	1,056
Lung	48	188	58	1,181	71	1,515
Prostate	93	1,097	66	845	50	551
Spleen	53	167	73	273	52	157
Total	351	4,894	290	4,414	239	3,565

FTUFinder localizes candidate units and supplies scale estimates

FTUFinder is the first model stage. It localizes candidate FTUs through bounding boxes, assigns confidence scores, and supplies object-size estimates for downstream resizing, where the size of a detected box of width w and height h is defined as $s = \sqrt{wh}$ in original-image pixels. The detector uses an RF-DETR detection-transformer framing in which a variable number of candidate boxes are predicted from a tiled histology image^{5,6}.

The scale-dynamic design is motivated by a simple imaging analogy. When photographing an object, the camera can zoom in, zoom out, or move closer so that the object occupies an appropriate size in the frame. FTURader translates this operation into microscopy: FTUFinder estimates the object location and size, the dynamic scaler adjusts apparent object scale and partitions large inputs into overlapping 1024×1024 -pixel tiles, and the mask decoder receives each FTU at a more consistent apparent scale. This design is intended to reduce the burden on a single

fixed input scale to cover small crypts, larger white-pulp regions, hollow alveoli, and other organ-specific FTU geometries.

Figure 5.2 focuses on this FTU detection and dynamic-tiling stage. Candidate patches are sampled on stain-normalized images after empty regions are filtered. A first FTUFinder pass raster-scans scout regions across the non-empty tissue area at native, zoom-in, and zoom-out resolutions to estimate the characteristic FTU size. The dynamic scaler then computes a resize factor from these candidate detections, rescales the full input image, partitions the scaled image into overlapping 1024×1024 -pixel tiles, and runs FTUFinder on each tile. Tile-level boxes and masks are mapped back to the original image coordinate system and merged with a SAHI-based duplicate-removal step implemented using the Roboflow Supervision library^{7,8}.

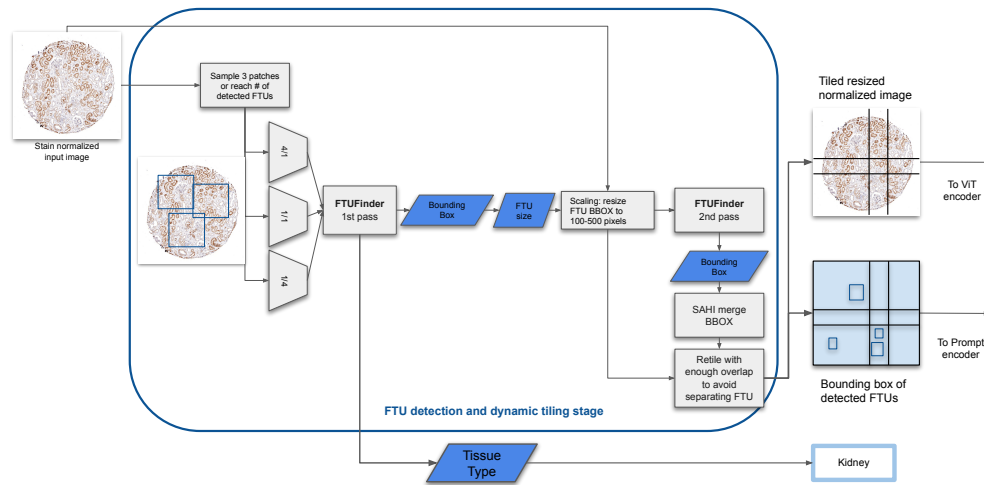


Figure 5.2: FTUFinder localization and scale-estimation stage. Candidate patches are sampled on stain-normalized images after empty regions are filtered. A first FTUFinder pass raster-scans scout regions across the non-empty tissue area at native, zoom-in, and zoom-out resolutions to estimate the characteristic FTU size. The dynamic scaler uses this estimate to resize the full image, partitions the scaled image into 1024×1024 -pixel tiles with 50% overlap, and runs FTUFinder again to generate final boxes for SAM2 mask decoding. SAHI-based merging, implemented using Roboflow Supervision, maps tile-level boxes and masks back to the whole image and removes duplicate predictions^{7,8}.

Box-prompted decoding converts detections into masks

The second-stage mask decoder follows the promptable segmentation logic of SAM and SAM2^{9,10}. For each candidate FTU, FTUFinder supplies a bounding-box prompt and the dynamic scaler supplies a tile at the selected apparent scale. The

mask decoder then predicts the FTU mask within that prompt. Each decoded mask is scored, filtered, and merged into the final output.

This design addresses a limitation of generic promptable segmentation on FTU images. Uniform grid prompts can waste prompts on empty lumen, background, or non-FTU tissue structures, whereas detector-derived boxes provide an anatomical prior about where the FTU is and how much context should be included. The detector prompt is therefore not only a computational convenience; it defines the candidate unit that the mask decoder should refine.

Evaluation prioritizes instance-level recovery and contextualizes pixel scores

Because FTURader is designed as an instance-segmentation pipeline, the primary evaluation question is whether it recovers separate FTU objects. Pixel-level Dice and IoU remain useful secondary summaries because they measure whether the union of predicted masks overlaps the annotated foreground. Both views are reported because semantic overlap can remain high even when biologically meaningful instances are merged, split, or duplicated.

Figure 5.3 summarizes object-level performance using F1 at fixed IoU thresholds and COCO AP metrics. Figure 5.4 isolates the detector stage by plotting FTUFinder bounding-box F1 across IoU thresholds, separating localization quality from downstream mask decoding. Figure 5.5 reports per-image Dice and IoU distributions after predicted instances are collapsed into a semantic foreground mask. Figure 5.6 compares semantic pixel scores with a reference Team 1 result from the original challenge context.

The Team 1 panel is retained as historical context rather than as primary evidence of FTURader quality. The Kaggle leaderboard setting rewarded semantic Dice optimization, whereas this chapter evaluates an instance-aware detector-prompted design. Public solution reports for the leading challenge entries describe extensive use of external image data, pseudo-labels, annotation regeneration, and task-specific curation^{3,11–13}. These strategies are valuable for leaderboard optimization but are not controlled baselines for evaluating whether detector prompting and dynamic scaling provide useful FTU instances. For this reason, pixel-level Dice values from leaderboard-optimized semantic models are treated as contextual references rather than direct one-to-one comparisons.

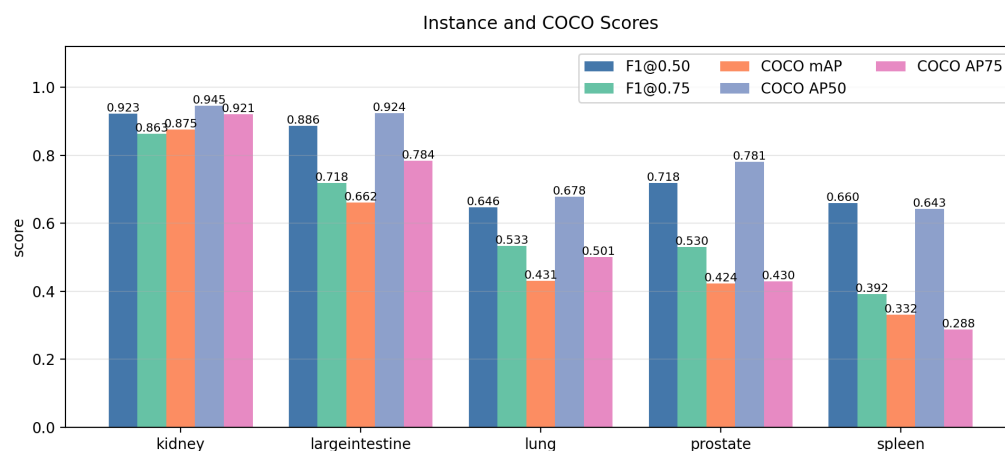


Figure 5.3: Instance and COCO scores for FTU segmentation across organ-specific FTUs. The plot reports object-level F1 at IoU thresholds of 0.50 and 0.75, together with COCO mAP, AP50, and AP75. Higher values indicate better recovery of separate FTU objects.

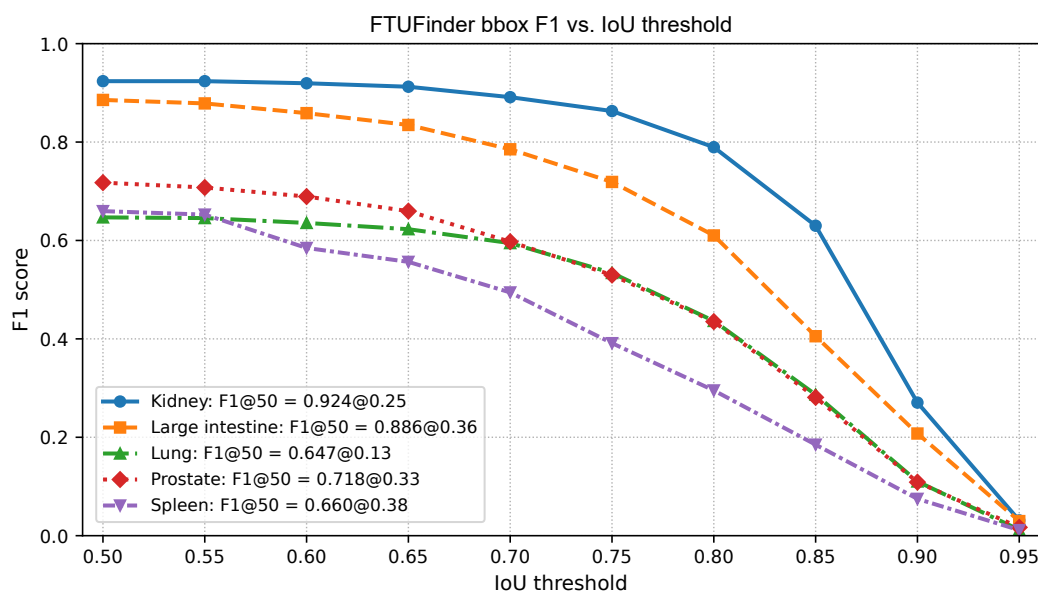


Figure 5.4: FTUFinder bounding-box F1 as a function of the IoU threshold used for matching predictions to ground truth. This detector-only view separates localization quality from mask-decoder quality and helps identify organs where downstream segmentation is limited by missed or imprecise boxes.

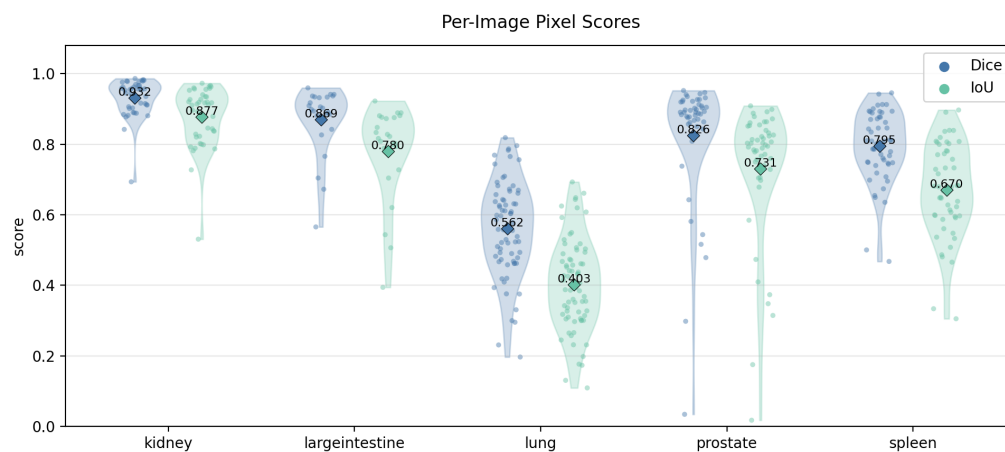


Figure 5.5: Per-image pixel-level Dice and IoU distributions for each organ-specific FTU category. Diamonds mark mean scores, while individual points show image-level variability. These semantic foreground metrics complement, but do not replace, instance-level object metrics.

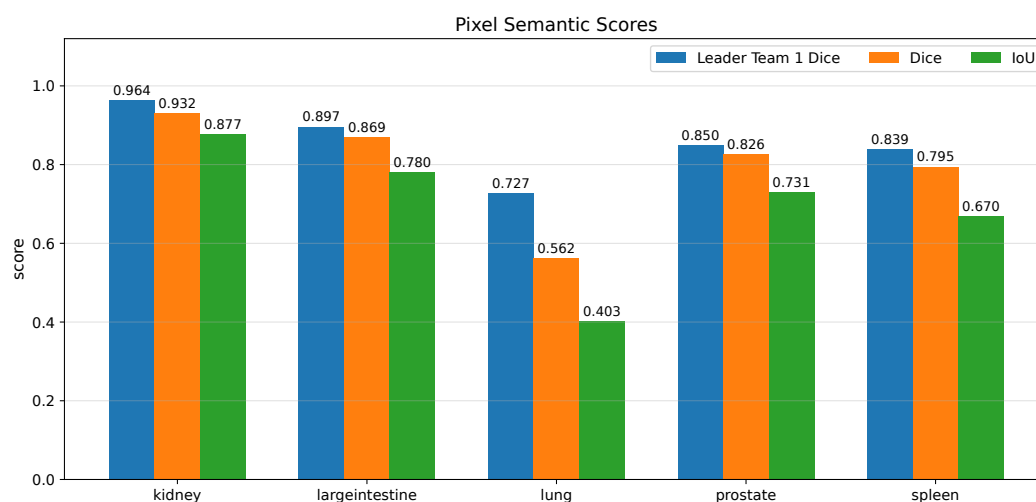


Figure 5.6: Pixel semantic scores compared with a Team 1 reference result from the original FTU challenge setting. For each organ, the figure shows the reference Dice score, the FTURader-derived semantic Dice score, and the difference. This panel is retained as historical context rather than primary evidence of instance-segmentation quality.

5.3 Discussion

This chapter reframes FTU segmentation as an instance-level problem. The central methodological idea is that FTUs should be localized, scale-normalized, and decoded as separate anatomical units rather than treated only as semantic foreground pixels. FTURader implements this idea through a detector-prompted pipeline: FTUFinder proposes object boxes and approximate scale, the dynamic scaler adapts the field of view, and a SAM2-based decoder produces one candidate mask per prompted FTU.

The evaluation examines candidate instance labels, FTUFinder localization, decoded masks, pixel-level semantic summaries, and detector training with and without dynamic scaling. The results support the use of object-aware evaluation for FTUs and show that the detector-prompted formulation can recover candidate FTU instances. Performance remains organ-dependent: compact and visually distinct FTUs, such as kidney glomeruli and large-intestine crypts, are more straightforward than ambiguous structures such as lung alveoli, prostate acini, and splenic white pulp, whose boundaries depend more strongly on context, scale, and tissue composition.

Several limitations remain. First, instance labels derived from semantic masks can inherit annotation ambiguity and may require expert review before they can support strong object-level claims. Second, organ-specific detectors and thresholds may reduce transferability to unseen FTU classes or new staining distributions. Third, dynamic scaling can introduce interpolation artifacts or lose context if crop padding and target size are not chosen carefully. Fourth, duplicate detections can reduce instance F1 even when semantic Dice improves. Finally, whole-slide deployment requires runtime, memory, and tile-merging analysis rather than only tile-level metrics.

Future work can build on the detector-prompted instance-segmentation framework evaluated here. Multiplex-channel processing may allow the pipeline to use additional histological and molecular information from H&E, PAS, IHC, and spatial-profiling images. Parameter-efficient adaptation, including LoRA-based fine-tuning, may reduce the cost of adding new organs or FTU classes. Language-guided prompting could support anatomically defined units that are not represented in the training set, provided that these definitions are paired with careful biological validation. Integrating cell-type and tissue-structure features from CellSAM and DeepCell-Type may be especially useful for ambiguous structures such as alveoli, where septal composition contributes to the object definition. Finally, super-resolution

and image-enhancement methods may improve segmentation when FTUs are small relative to the input crop, but these modules should be evaluated for segmentation accuracy, runtime, and robustness before inclusion in a deployed workflow^{14–16}.

Overall, FTURader demonstrates a practical route from semantic FTU foreground masks to detector-prompted FTU instances. The chapter establishes dynamic scaling and box-prompted decoding as a framework for multi-organ FTU segmentation, while also showing why FTU analysis is more comprehensively evaluated with object-level metrics than with pixel overlap alone.

5.4 Methods

Dataset sources and splits

The dataset is derived from the HuBMAP/HPA FTU segmentation setting and includes organ-associated FTU categories used in the Hacking the Human Body challenge³. The target FTU categories are kidney glomeruli, large-intestine crypts, lung alveoli, prostate glandular acini, and splenic white pulp. We followed the official data split: the public training set was used for model training, the official public-test split was used internally for validation and threshold selection, and the private test set was reserved for final held-out testing. Source images and masks were organized by organ, data provider, staining or imaging protocol when available, and official split. Organ-level image and annotation counts are reported in Table 5.1.

Source semantic masks were converted into candidate instance annotations before detector training using the official OpenCV-based mask-transcoding script from the Hacking the Human Body organ-segmentation competition³. The conversion assigns a unique positive integer label to each candidate FTU against a zero-valued background. Candidate masks were filtered for invalid geometry, converted into bounding boxes and segmentation records, and exported in COCO format for FTUFinder training and evaluation¹⁷. The same candidate instances were also exported in an SA-1B-compatible format for SAM2 mask-decoder training, preserving image metadata, box prompts, and mask targets in the format introduced with SAM⁹. Because this transcoding is the official conversion released by the challenge dataset providers, the resulting instance annotations are used as the operational FTU instance reference for this study rather than re-adjudicated here.

Image preprocessing and stain normalization

Images were preprocessed before detection and mask decoding to reduce empty-region sampling and stain variation. The final model inputs used static Macenko-

normalized RGB images. The normalizer was fit to a selected reference H&E image with `torchstain`, and each source image was mapped toward that reference stain distribution before detector or mask-decoder input^{18,19}. Macenko stain separation was also used to generate hematoxylin-like and eosin-like channels for visualization and quality control rather than as required model inputs. This quality-control check follows the biological basis of H&E staining, in which hematoxylin highlights nuclear material and eosin counterstains cytoplasmic and extracellular structures²⁰. Reinhard color transfer, color deconvolution, and sparse stain-separation methods provide related background for this class of preprocessing^{21–23}.

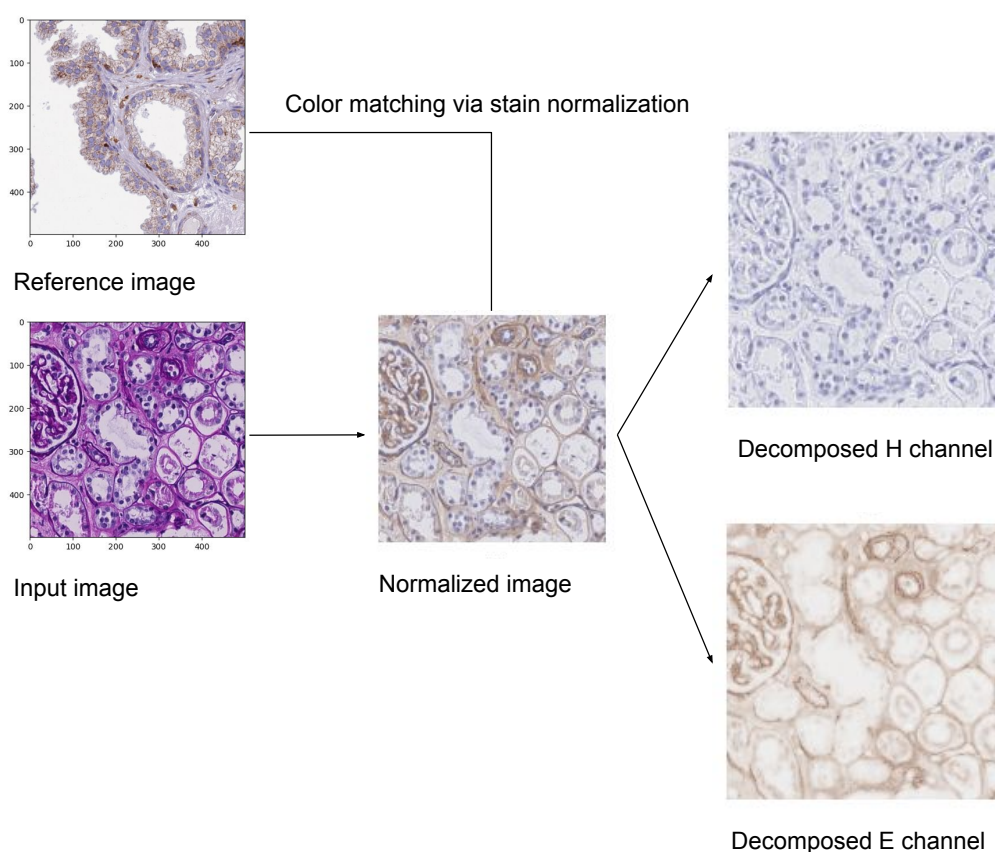


Figure 5.7: Macenko stain normalization and H/E channel decomposition for FTU preprocessing. The workflow uses `torchstain` to map heterogeneous H&E images toward a selected reference appearance and then decomposes the normalized image into hematoxylin-like and eosin-like channels for visualization and quality control. Successful normalization should preserve interpretable stain-specific structure: nuclei should remain prominent in the hematoxylin-like channel, while cytoplasm and extracellular matrix should remain prominent in the eosin-like channel.

FTUFinder architecture and training

FTUFinder was trained as an organ-specialist FTU detector over tiled or resized histology images. It predicts candidate bounding boxes, class labels, and confidence scores. The detector used an RF-DETR Large architecture in a DETR-based set-prediction framing^{5,6}. Detector outputs were used both for FTU localization and for estimating object scale before mask decoding.

The training augmentation pipeline used static stain normalization before geometric and color perturbations. Each training image was first normalized to the chosen reference image. The normalized image was then randomly resized across a range from 4× downsampling to 4× upscaling, randomly cropped to match the model input size, and augmented with random horizontal and vertical flipping, random rotation, and `RandomPerspective` transforms. Random color jitter was applied after stain normalization and perturbed brightness, contrast, saturation, hue, and related color properties.

Five specialist FTUFinder models were trained, one for each organ-associated FTU category. This organ-specialist binary-detector setup was chosen because the multi-organ FTU benchmark reported that binary models can outperform a single multi-class model for FTU segmentation across organs³. Each specialist detector was trained on a single NVIDIA RTX A6000 GPU, with training time ranging from 35 to 41 minutes per organ. The final detector used 300 object queries and selected up to 100 predictions per image, a smaller query budget than the 3500-query dense detection setting used in CellSAM⁴. Table 5.2 summarizes the shared training hyperparameters.

Figure 5.8 shows the FTUFinder training loss and validation F1 curves for the dynamic-scaling training setup. Figure 5.9 shows the validation F1 curve when dynamic scaling is removed; the lower validation F1 supports the use of scale-normalized training for FTU detection.

Dynamic scaling and tile merging

The dynamic scaler uses an explicit scout-and-rescale procedure to make the apparent FTU size more consistent before mask decoding. For each input image, scout patches are placed over the non-empty tissue region by a deterministic raster scan: non-overlapping 1024×1024 patches are tiled left to right and top to bottom, and only the upper-left corner of the first patch is randomized (drawn from a 50×50 -pixel box at the top-left of the tissue region), so that all remaining patch positions follow

Table 5.2: FTUFinder training hyperparameters. The same RF-DETR Large configuration was used for the five organ-specialist detectors.

Hyperparameter	Value
<code>batch_size</code>	16
<code>grad_accum_steps</code>	1
Epochs	100
Learning rate	1×10^{-4}
<code>lr_encoder</code>	1.5×10^{-4}
<code>lr_scheduler</code>	step
<code>weight_decay</code>	1×10^{-4}
Optimizer	AdamW
Base model	RF-DETR Large
<code>num_queries</code>	300
<code>num_select</code>	100
<code>amp</code>	True
Resolution	1024×1024

deterministically from this origin. Each scout patch is evaluated at three resolution layers: (i) the native $1 \times 1024 \times 1024$ patch, (ii) a $4 \times$ zoom-in layer formed by dividing the same field of view into sixteen 256×256 subtiles and resizing each subtile to 1024×1024 , and (iii) a $4 \times$ zoom-out context crop of 4096×4096 pixels resized to 1024×1024 . If the image is smaller than the zoom-out crop, the full image is used instead. All detections from this scout stage are transformed back to the original image coordinate system before size statistics are computed. The scan starts with three scout patches and, if fewer than ten FTU detections have been accumulated, continues to additional raster patches until at least ten detections are collected or the tissue region is fully covered. Because the same FTU can be detected more than once across the three resolution layers and adjacent raster patches, these scout detections are merged by non-maximum suppression using an intersection-over-smaller-area (IoS) threshold of 0.6 so that each FTU contributes only once to $\bar{s}$.

For a detected bounding box with width w and height h in original-image pixels, FTU size is defined as

$$s = \sqrt{wh}. \quad (5.1)$$

The image-level characteristic size is the arithmetic mean of the pooled scout-box sizes, $\bar{s}$. The dynamic resize factor is then

$$\alpha = \frac{300}{\bar{s}}, \quad (5.2)$$

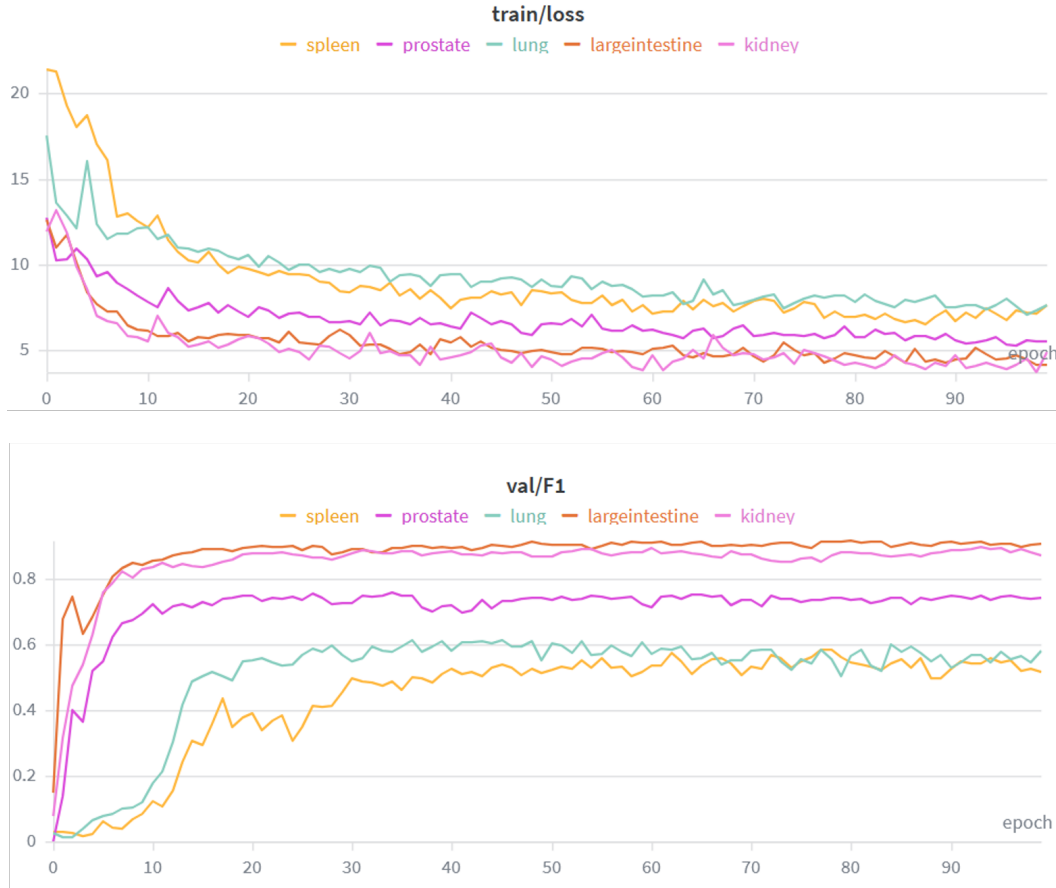


Figure 5.8: FTUFinder training curves with dynamic scaling. The plot reports training loss and validation F1 during training of the organ-specialist FTU detectors. These curves summarize convergence for the detector stage before box prompts are passed to the SAM2-based mask decoder.

so that the average detected FTU is presented to the downstream detector and mask decoder at approximately 300 pixels. The target operating range is 100–500 pixels. As an implementation diagnostic, a warning is recorded when the resized scout detections would still contain objects smaller than 50 pixels or larger than 700 pixels, that is, when $\alpha s_{\min} < 50$ or $\alpha s_{\max} > 700$.

If the raster scan covers the entire tissue region without producing any detections, the image is treated as containing no candidate FTUs and the pipeline returns an empty prediction set. Otherwise, the full input image is resized by α and partitioned into 1024×1024 tiles with 50% overlap. FTUFinder is applied to each scaled tile, and each retained box is passed to the SAM2-based mask decoder to generate an FTU mask. Tile-level predictions are converted back to the original

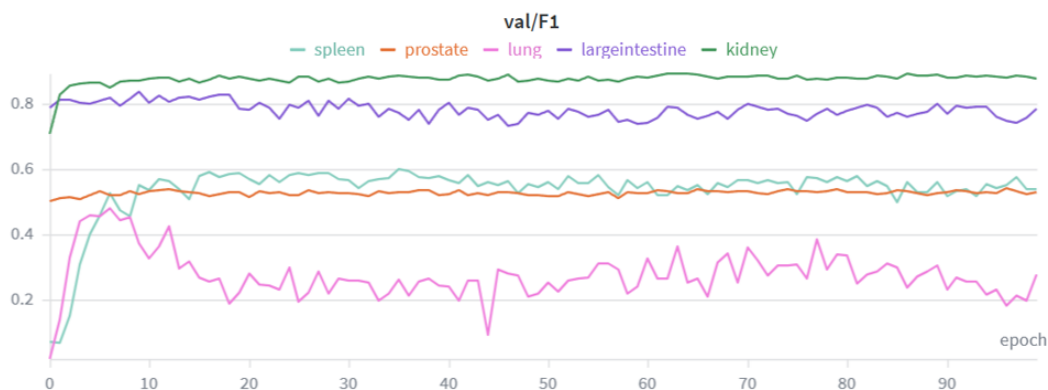


Figure 5.9: FTUFinder validation F1 without dynamic scaling. Removing dynamic scaling reduces validation F1, indicating that scale normalization is important for presenting heterogeneous FTU sizes to the detector.

image coordinates by adding the tile offset and applying the inverse scale factor $1/\alpha$; masks are inverse-resized and cropped to the original image extent. Finally, SAHI non-maximum merging, implemented with the Roboflow Supervision library, consolidates duplicate or highly overlapping boxes and masks using an overlap threshold of 0.6^{7,8}.

SAM2-based mask decoder training

The mask decoder receives a fixed-size image crop and is trained through the SAM2 fine-tuning pipeline. During training, each prompt is paired with the candidate instance mask corresponding to that FTU. The decoder follows the SAM/SAM2 promptable segmentation family^{9,10}. We trained one generalist SAM2-based mask decoder across all five organ-associated FTU categories, while localization remained handled by the five organ-specialist FTUFinder models. The generalist SAM2 run used the same training-augmentation pipeline as FTUFinder, including static stain normalization, random resize, random crop, flipping, rotation, RandomPerspective, and post-normalization color jitter. The model was trained on a single NVIDIA RTX A6000 GPU for 10 hours and 41 minutes. A batch size of 1 was used to fit the 48 GB GPU memory budget, `num_frames` was set to 1 because the task uses single images rather than video sequences, `max_num_objects` was set to 5 as a training-time memory and runtime guard, and automatic mixed precision was enabled for faster training. Table 5.3 summarizes the SAM2 training hyperparameters.

Importantly, `max_num_objects=5` was not an inference-time cap on the number of FTUs in an image or tile. It only limited how many object prompts and targets were randomly sampled simultaneously in one training forward pass, because the number of FTUs in a tile is variable and dense large-intestine or lung tiles can exceed GPU memory or substantially slow training if all objects are loaded at once. At inference time, the number of masks is determined by FTUFinder: every retained bounding box is used as a prompt for the SAM2-based decoder. If many boxes are present, prompts can be processed sequentially or in implementation-level batches for memory efficiency, and the resulting masks are merged back into image coordinates; no limit of five FTUs per tile or per image is imposed during inference.

Table 5.3: SAM2-based mask-decoder training hyperparameters. A single generalist model was trained across all five organ-associated FTU categories using the same augmentation pipeline as FTUFinder training. The `max_num_objects` value is a training-time memory limit and does not cap the number of inference prompts.

Hyperparameter	Value
<code>batch_size</code>	1
<code>num_frames</code>	1
Epochs	300
<code>base_lr</code>	5.0×10^{-6}
<code>vision_lr</code>	3.0×10^{-6}
Optimizer	AdamW
Base model	<code>sam2.1_hiera_base_plus</code>
<code>max_num_objects</code>	5 (training only)
<code>amp</code>	True
Resolution	1024×1024

Figure 5.10 shows the training-loss curve for the 300-epoch generalist SAM2 training run.

Inference and threshold optimization

During inference, each organ-specific FTUFinder produces candidate boxes and bounding-box confidence scores. The validation-selected threshold is applied only to these FTUFinder box scores to determine which boxes are retained for mask decoding. Thresholding on the box score rather than on a mask-derived score is appropriate because each retained box maps one-to-one to a decoded mask, decoding masks for very-low-confidence boxes is wasteful, and sweeping a mask-score threshold would require decoding every low-score box (on the order of 300 per 1024×1024 crop at a 0.05 cutoff), which is computationally prohibitive. For each



Figure 5.10: SAM2-based mask-decoder training loss. The curve shows loss over the 300-epoch generalist training run across the five organ-associated FTU categories. The model uses FTUFinder box prompts and local image context to predict FTU instance masks.

specialist FTUFinder model, we swept box-score thresholds from 0.05 to 0.99 on the validation set. At each threshold, predicted boxes were matched to ground-truth boxes using $\text{IoU} = 0.5$. Precision, recall, and F1 were computed from true positives, false positives, and false negatives, and the threshold that maximized bounding-box F1 was selected for that organ.

After thresholding and mask decoding, each retained prediction is assigned a final confidence estimate. This final confidence is not used for threshold optimization; it is reported with the output mask to summarize both detection confidence and mask-decoder confidence. We define it as the geometric mean of the FTUFinder box score and the SAM2-based mask score:

$$c_{\text{mask}} = \sqrt{c_{\text{FTUFinder}} \times c_{\text{SAM}}}, \quad (5.3)$$

where $c_{\text{FTUFinder}}$ is the FTUFinder bounding-box confidence after thresholding and c_{SAM} is the mask-decoder confidence for the corresponding prompted mask. This combination is a heuristic confidence estimate rather than a calibrated probability, and because each specialist FTUFinder detector handles a single organ, predictions are ranked only within that organ; c_{mask} is therefore not used for any cross-organ score comparison.

Table 5.4 reports the selected bounding-box score threshold for each specialist FTUFinder model. These organ-specific box-score thresholds were fixed after validation and used to select prompts for downstream mask decoding and final test-set evaluation.

Table 5.4: Validation-selected FTUFinder score thresholds. Thresholds were selected independently for each organ-specialist detector by sweeping score thresholds from 0.05 to 0.99 and maximizing bounding-box F1 at IoU = 0.5 on the validation set.

Organ	Score threshold
Kidney	0.25
Large intestine	0.36
Lung	0.13
Prostate	0.33
Spleen	0.38

Threshold choice controls a trade-off between object-level and pixel-level metrics. Lower thresholds may retain duplicate detections, decreasing instance-level F1 by adding false positives, while potentially improving semantic foreground coverage after mask aggregation. Higher thresholds may reduce duplicate predictions but miss difficult FTUs. For this reason, detector thresholds were selected with an explicit instance-level criterion and then fixed before final testing. It is important to note, however, that an F1-optimized threshold is not necessarily optimal for human-facing review or deployment. Because F1 rewards recall as well as precision, the selected threshold can retain additional candidate boxes in difficult categories such as lung and prostate. In practical applications where cleaner visual outputs are preferred, users may increase the bounding-box score threshold to obtain more conservative predictions, accepting a potential reduction in recall.

Evaluation metrics

The evaluation is organized into bounding-box-level, instance-level, and pixel-level measurements. All metrics were computed separately for each organ-specific FTU category. Bounding-box evaluation matches predicted and ground-truth boxes by IoU. For a selected IoU threshold, matched predictions are counted as true-positive objects, unmatched predictions as false-positive objects, and unmatched ground-truth boxes as false-negative objects. Precision, recall, and object-level F1 are then

computed from these object counts:

$$F1_{\text{object}} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \quad (5.4)$$

Instance-level segmentation is evaluated after converting each predicted FTU into an individual mask instance. Predicted and ground-truth instances are matched by mask IoU. COCO AP summarizes precision across IoU thresholds from 0.50 to 0.95 in increments of 0.05, while COCO AP50 and AP75 report AP at fixed IoU thresholds of 0.50 and 0.75¹⁷. COCO AP ranks predictions by the combined confidence $c_{\text{mask}} = \sqrt{c_{\text{FTUFinder}} \times c_{\text{SAM}}}$. Because average precision integrates over the full precision–recall curve rather than scoring a single operating point, the AP computation retains FTUFinder boxes at a low score threshold of 0.05, instead of the higher validation-selected per-organ thresholds (Table 5.4), so that lower-confidence detections still contribute to the ranking. These object-level metrics evaluate whether the pipeline identifies separate FTU objects rather than only predicting the correct foreground pixels.

Pixel-level evaluation collapses predicted FTU instances into a semantic foreground mask and compares it with the ground-truth foreground mask. Pixel-level foreground Dice and IoU are computed per image as

$$\text{Dice}_{\text{pixel}} = \frac{2|P \cap G|}{|P| + |G|}, \quad \text{IoU}_{\text{pixel}} = \frac{|P \cap G|}{|P \cup G|}, \quad (5.5)$$

where P is the predicted foreground pixel set and G is the ground-truth foreground pixel set. Equivalently, when true positives (TP_{pixel}), false positives (FP_{pixel}), and false negatives (FN_{pixel}) denote foreground pixels rather than matched instances, as discussed in Section 2.3,

$$\text{Dice}_{\text{pixel}} = \frac{2TP_{\text{pixel}}}{2TP_{\text{pixel}} + FP_{\text{pixel}} + FN_{\text{pixel}}}. \quad (5.6)$$

$$\text{IoU}_{\text{pixel}} = \frac{TP_{\text{pixel}}}{TP_{\text{pixel}} + FP_{\text{pixel}} + FN_{\text{pixel}}}. \quad (5.7)$$

Code availability

The FTURader codebase is maintained in a private GitHub repository at https://github.com/vanvalenlab/ftu_dev. A public repository will be released after publication.

5.5 Extended Data

Qualitative visualizations show pixel-level and instance-level outputs

Qualitative inspection complements the quantitative metrics by showing whether predicted foreground pixels also form plausible individual FTU instances. For each organ-specific FTU category, five private-test images were randomly selected for qualitative review. The examples shown in Figures 5.11–5.15 were chosen from those randomly selected images and are not intended to represent either the best or the worst cases. Each visualization includes both pixel-level semantic results and instance-level predictions so that foreground coverage and object separation can be assessed together.

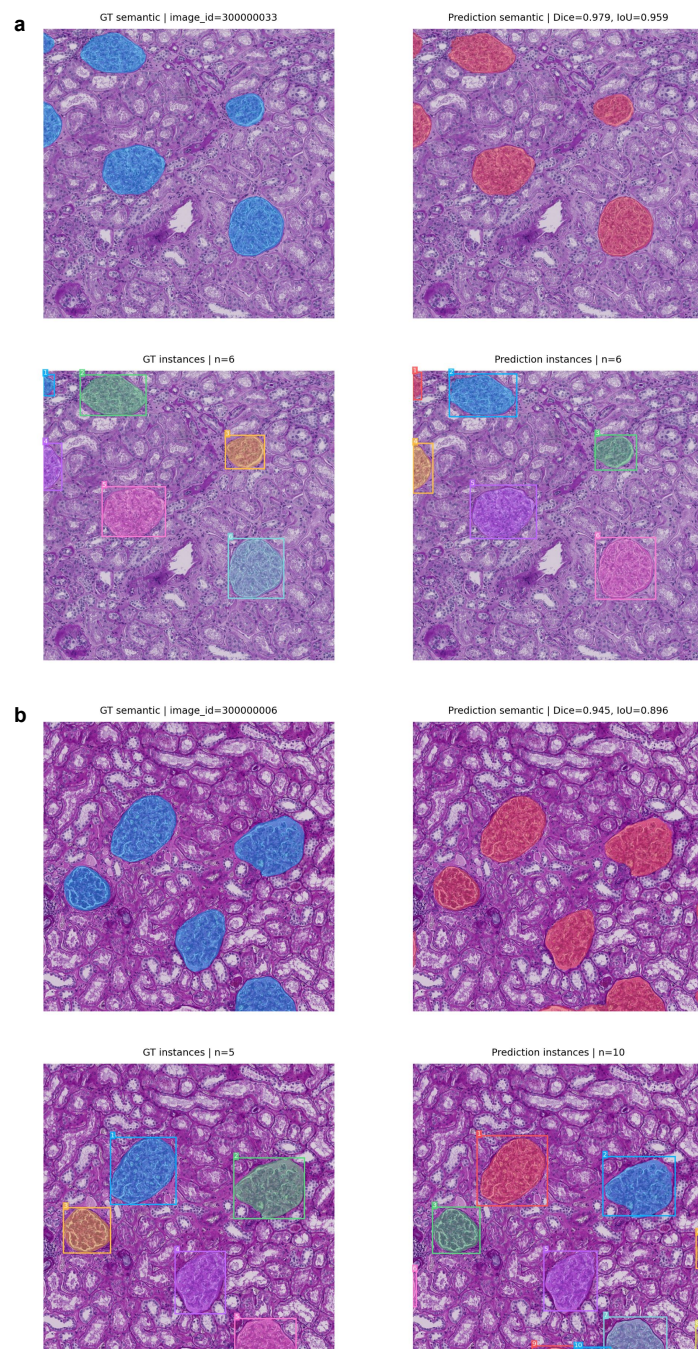


Figure 5.11: Kidney private-test examples. Panels **a** and **b** show two images with pixel-level semantic and instance-level glomerulus results. The examples were chosen from five randomly selected private-test images and are not intended to show best-case or worst-case performance.

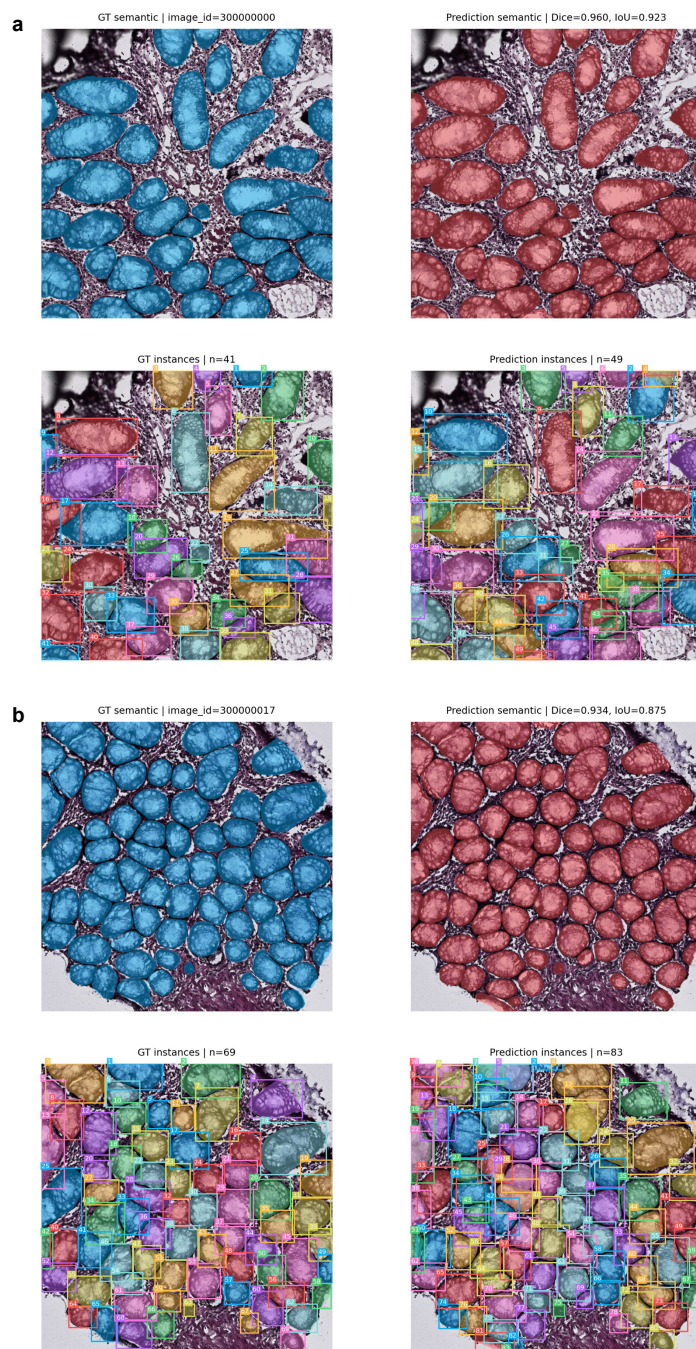


Figure 5.12: Large-intestine private-test examples. Panels **a** and **b** show two images with pixel-level semantic and instance-level crypt results. The examples were chosen from five randomly selected private-test images and are not intended to show best-case or worst-case performance.

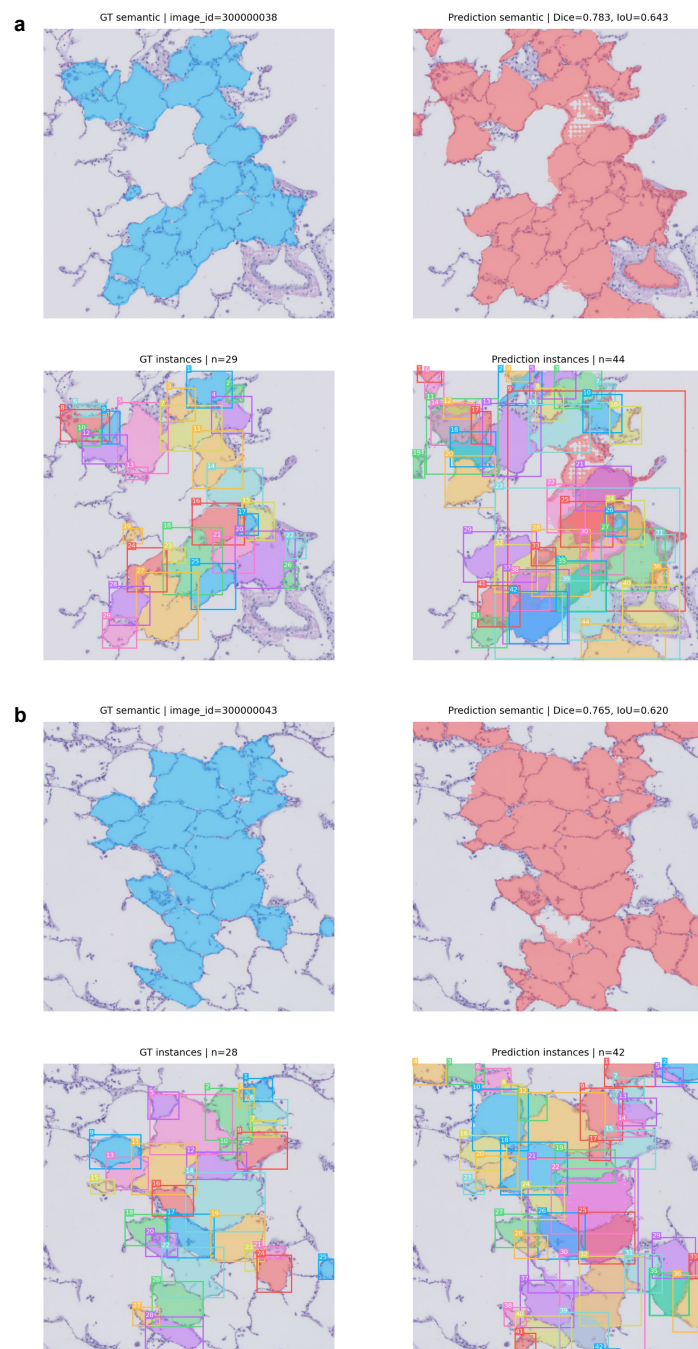


Figure 5.13: Lung private-test examples. Panels **a** and **b** show two images with pixel-level semantic and instance-level alveolus results. The examples were chosen from five randomly selected private-test images and are not intended to show best-case or worst-case performance.

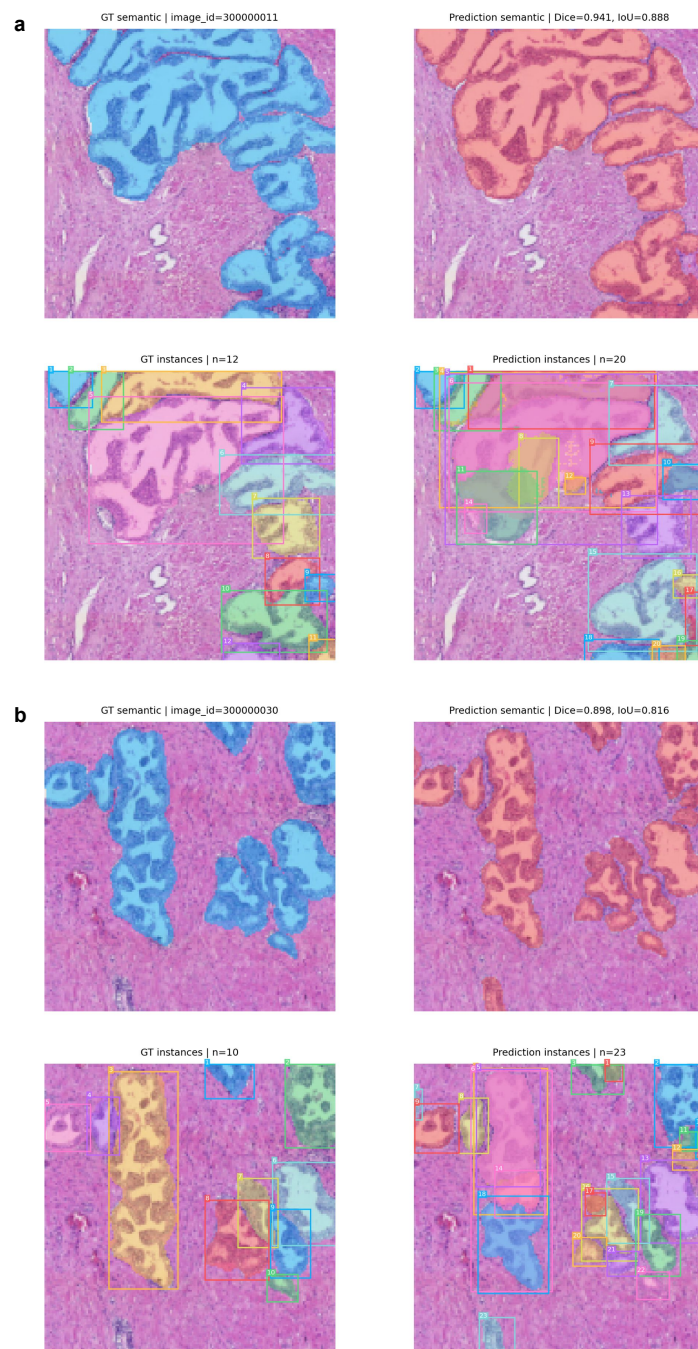


Figure 5.14: Prostate private-test examples. Panels **a** and **b** show two images with pixel-level semantic and instance-level glandular acinus results. The examples were chosen from five randomly selected private-test images and are not intended to show best-case or worst-case performance.

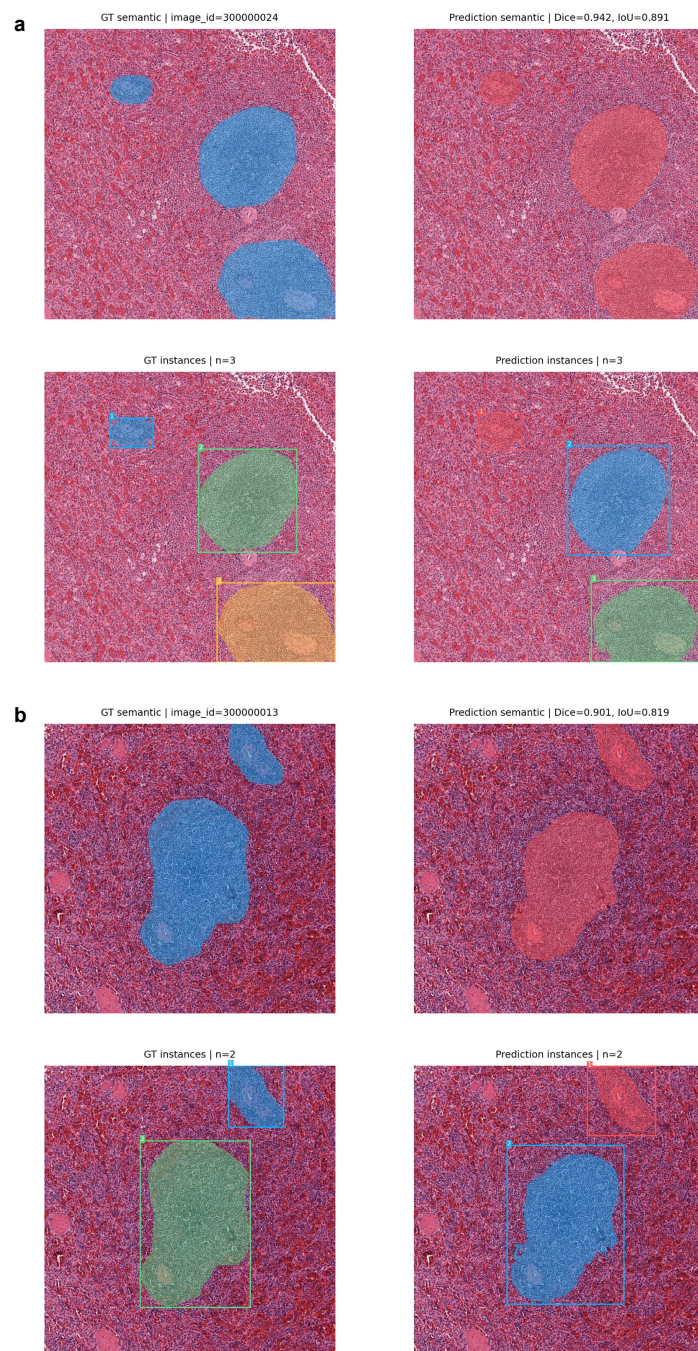


Figure 5.15: Spleen private-test examples. Panels **a** and **b** show two images with pixel-level semantic and instance-level white-pulp results. The examples were chosen from five randomly selected private-test images and are not intended to show best-case or worst-case performance.

References

1. HuBMAP Consortium. “The human body at cellular resolution: the NIH Human Biomolecular Atlas Program”. In: *Nature* 574.7777 (Oct. 9, 2019), pp. 187–192. doi: 10.1038/s41586-019-1629-x.
2. Yashvardhan Jain et al. “Segmentation of human functional tissue units in support of a Human Reference Atlas”. In: *Communications Biology* 6.1 (July 19, 2023), p. 717. doi: 10.1038/s42003-023-04848-5.
3. Yashvardhan Jain et al. “Segmenting functional tissue units across human organs using community-driven development of generalizable machine learning algorithms”. In: *Nature Communications* 14.1 (Aug. 3, 2023), p. 4656. doi: 10.1038/s41467-023-40291-0.
4. Markus Marks et al. “CellSAM: a foundation model for cell segmentation”. In: *Nature Methods* 22.12 (Dec. 8, 2025), pp. 2585–2593. doi: 10.1038/s41592-025-02879-w.
5. Isaac Robinson et al. “RF-DETR: Neural Architecture Search for Real-Time Detection Transformers”. In: *International Conference on Learning Representations*. 2026. arXiv: 2511.09554.
6. Nicolas Carion et al. “End-to-End Object Detection with Transformers”. In: *Computer Vision – ECCV 2020*. Springer, 2020, pp. 213–229. doi: 10.1007/978-3-030-58452-8_13. arXiv: 2005.12872.
7. Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. “Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection”. In: *2022 IEEE International Conference on Image Processing (ICIP)*. 2022, pp. 966–970. doi: 10.1109/icip46576.2022.9897990. arXiv: 2202.06934.
8. Roboflow. *Supervision: Reusable Computer Vision Tools*. Open-source Python library for building computer vision applications. 2024. URL: <https://github.com/roboflow/supervision>.
9. Alexander Kirillov et al. “Segment Anything”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4015–4026. arXiv: 2304.02643.
10. Nikhila Ravi et al. “SAM 2: Segment Anything in Images and Videos”. In: *International Conference on Learning Representations*. 2025. arXiv: 2408.00714.
11. Opusen. *I Need SegFormer: 1st Place Solution*. Kaggle. 2022. URL: <https://www.kaggle.com/competitions/hubmap-organ-segmentation/writeups/i-need-segformer-1st-place-solution>.
12. Victor Durnov. *Victor Durnov: 2nd Place Solution*. Kaggle. 2022. URL: <https://www.kaggle.com/competitions/hubmap-organ-segmentation/writeups/victor-durnov-2nd-place-solution>.

13. Volodymyr Sydorskyi et al. “Semi-Supervised Segmentation of Functional Tissue Units at the Cellular Level”. In: *Proceedings of the CLEF 2022 Labs and Workshops*. Vol. 3384. CEUR Workshop Proceedings. 2023, pp. 12–27. arXiv: 2305.02148.
14. Xiaoliao Xu, Saarthak Kapse, and Prateek Prasanna. “SuperDiff: A diffusion super-resolution method for digital pathology with comprehensive quality assessment”. In: *Medical Image Analysis* 106 (Jan. 2026), p. 103808. doi: 10.1016/j.media.2025.103808.
15. Yijie Zhang et al. “Pixel super-resolved virtual staining of label-free tissue using diffusion models”. In: *Nature Communications* 16.1 (May 30, 2025), p. 5016. doi: 10.1038/s41467-025-60387-z.
16. Ziyi Liu et al. “A unified low-level foundation model for enhancing pathology image quality”. In: *arXiv preprint* (Sept. 1, 2025). doi: 10.48550/arxiv.2509.01071. arXiv: 2509.01071.
17. Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Springer, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48. arXiv: 1405.0312.
18. Marc Macenko et al. “A method for normalizing histology slides for quantitative analysis”. In: *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. 2009, pp. 1107–1110. doi: 10.1109/isbi.2009.5193250.
19. Carlo Alberto Barbano and André Pedersen. *EIDOSLAB/torchstain: v1.2.0-stable*. Version v1.2.0-stable. Aug. 12, 2022. doi: 10.5281/zenodo.6979540.
20. Andrew H. Fischer et al. “Hematoxylin and Eosin Staining of Tissue and Cell Sections”. In: *Cold Spring Harbor Protocols* 2008.5 (2008), pdb.prot4986. doi: 10.1101/pdb.prot4986.
21. Erik Reinhard et al. “Color transfer between images”. In: *IEEE Computer Graphics and Applications* 21.5 (Sept. 2001), pp. 34–41. doi: 10.1109/38.946629.
22. Arnout C. Ruifrok and Dennis A. Johnston. “Quantification of histochemical staining by color deconvolution”. In: *Analytical and Quantitative Cytology and Histology* 23.4 (2001), pp. 291–299. URL: <https://pubmed.ncbi.nlm.nih.gov/11531144/>.
23. Abhishek Vahadane et al. “Structure-Preserving Color Normalization and Sparse Stain Separation for Histological Images”. In: *IEEE Transactions on Medical Imaging* 35.8 (2016), pp. 1962–1971. doi: 10.1109/tmi.2016.2529665.

CONCLUSION AND FUTURE DIRECTIONS

6.1 Thesis Synthesis

Biological image segmentation is not only a prediction task. It is the measurement layer that turns microscopy images into biological objects that can be counted, compared, tracked, and interpreted. A cell boundary determines measured morphology and molecular neighborhood. A functional tissue unit (FTU) boundary determines which cells and extracellular regions are assigned to the same anatomical structure. Errors in these boundaries therefore propagate into biological conclusions about density, organization, spatial context, and disease-associated tissue states.

This thesis addressed segmentation as a multiscale measurement problem. The central claim is that reliable biological segmentation requires four ingredients: data representations that preserve the object being measured, evaluation protocols that match the biological question, model designs that fit the available annotation regime, and scale-aware adaptation when the target shifts from cells to tissue structures. The chapters test this claim across the progression from cell masks to FTU instances.

The progression mirrors a broader change in the field. U-Net¹, Mask R-CNN², DETR³, and Mask2Former⁴ established powerful architectures for semantic, instance, and panoptic segmentation. Cellpose⁵, StarDist⁶, Mesmer⁷, and Omnipose⁸ showed that biological shape, boundary, and flow priors can make cell segmentation practical across important microscopy settings. SAM⁹ and SAM2¹⁰ changed the interface by making segmentation promptable. This thesis shows that those advances become biologically useful only when the interface, data, and metrics are adapted to the object of interest.

Chapter 2 built the measurement foundation. Chapter 3 tested a geometric prior for limited-label cell segmentation. Chapter 4 moved to detector-guided generalist cell segmentation through CellSAM and CellFinder¹¹. Chapter 5 extended the same logic to FTUs through FTURader and FTUFinder, where scale, anatomy, tissue context, object size, and whole-slide constraints become central. Together, these chapters support a single conclusion: the biological unit being measured should determine the segmentation system, from data representation and evaluation metrics to model design.

6.2 Contributions Across Chapters

Chapter 2 established the data and evaluation foundation for the rest of the thesis. Heterogeneous cell segmentation datasets differ in image size, modality, annotation convention, mask representation, and evaluation metric. Without controlled representation and evaluation, improvements can be difficult to interpret because conversion, tiling, small-object encoding, or metric choice can change the apparent performance of a model. The chapter’s contribution is not a single all-purpose file format, but an explicit workflow for COCO-compatible conversion, small-object mask fidelity, dataset provenance, tiling, normalization, and object-level as well as pixel-level metric reporting^{12,13}.

Chapter 3 addressed the limited-label regime, where expert annotations are scarce and large models can be difficult to train from scratch. Microscopy images often lack a canonical in-plane orientation, making rotation and reflection equivariance a plausible prior¹⁴. Building on earlier evidence that rotation equivariance can improve sample efficiency in segmentation¹⁵, the chapter tested finite rotation/reflection-equivariant feature processing in a Mask2Former-based pipeline. The result was conditional: equivariance was most useful in the small *C. elegans* no-augmentation setting without pretraining, but it was not clearly superior on the larger TissueNet experiment and carried substantial runtime and memory cost.

Chapter 4 shifted from hand-designed priors toward foundation-model adaptation. Segment Anything⁹ demonstrated the power of promptable segmentation, but dense cell images pose a different problem from ordinary natural-image prompting. Cells are small, numerous, and embedded in modality-specific imaging characteristics. CellSAM addresses this mismatch by coupling CellFinder, which detects candidate cells, with a promptable mask decoder¹¹. The key design move is to make prompting automatic and biological: CellFinder supplies the cell hypotheses that make a general mask decoder useful for dense microscopy.

Chapter 5 extended this detector-prompted logic from cells to FTUs. FTUs are not merely larger cells. They are multicellular anatomical structures whose boundaries can depend on tissue context, lumen shape, organ type, and staining variation. Work from human reference atlas (HRA) construction has made these structures important targets for computational pathology and spatial biology^{16,17}. The chapter therefore proposed FTURader, a detector-first, scale-dynamic pipeline in which FTUFinder localizes candidate FTUs, a dynamic scaler normalizes their apparent size, and a SAM2-based decoder produces masks from prompts^{10,18}. The contribution is

a framework for moving from cell-level segmentation toward FTU segmentation without assuming that a cell model will transfer directly to anatomy.

6.3 Cross-Cutting Principles

The first principle is that evaluation must follow the biological question. Object-level metrics (F1, Jaccard, COCO AP) and pixel-level metrics (Dice, IoU) summarize different errors. A high pixel-overlap score can hide merged cells, split FTUs, or missed rare objects. Conversely, a model may identify most objects but still produce boundaries that are not sufficient for morphology or downstream spatial analysis. For that reason, evaluation should be interpreted as part of the experimental design rather than a final bookkeeping step¹³.

The second principle is that model bias and data scale are substitutes only within limits. When data are scarce, a reliable prior such as rotation/reflection equivariance can reduce the burden on labels and augmentation. When data diversity, pretraining, and promptable interfaces become available, a broader generalist model may cover visual variation that a narrow hand-coded prior cannot. This tradeoff explains the thesis movement from equivariant Mask2Former in Chapter 3 to CellSAM in Chapter 4.

The third principle is that promptable models need biological front ends. In this thesis, CellFinder supplies cell locations for CellSAM, and FTUFinder supplies tissue-unit locations and scale estimates for FTURader. This division separates the question of finding the biological object from the question of drawing its boundary. It also defines a failure path: missed or imprecise detections can propagate into the final masks. The practical value of promptable segmentation therefore depends on the detection quality of the prompt generator.

6.4 Limitations

The conclusions of this thesis are bounded by dataset coverage. Even large cell datasets cannot cover all species, instruments, stains, perturbations, disease states, or multiplexed assays. FTU datasets are narrower still because anatomical structures differ across organs and expert annotation is more specialized. Broad performance should therefore be interpreted as evidence across the tested distribution, not as task-independent reliability.

Annotation quality remains a second boundary. Segmentation models inherit the conventions and ambiguities of their labels. A mask can be internally consistent

yet biologically partial if annotators disagree about weak boundaries, overlapping objects, luminal spaces, or tissue-unit extent. Chapter 2 addressed representation fidelity, but representation fidelity is not the same as biological truth. Future benchmarks should report annotation lineage, review protocol, uncertainty, and expert adjudication when the masks will support biological claims.

Foundation models also introduce a third boundary. Their performance depends on training-data support, prompt reliability, and deployment-domain similarity. CellSAM provides strong evidence that detector-guided prompting can work across diverse cell images¹¹, but it still depends on robust detection and should be validated when applied to new modalities or unusual morphologies¹⁹. SAM-based models similarly require careful adaptation before they can be trusted in automatic biological workflows²⁰. In this sense, generality is an empirical property to be tested, not a label that removes the need for validation.

The FTU work has a different boundary. FTURader provides a detector-prompted route from semantic FTU foreground masks to candidate instances, but broad biological claims require stronger validation across organs, scanners, stains, tissue-preparation protocols, disease states, and expert review workflows. The biological utility of FTU segmentation should ultimately be judged by whether it strengthens tissue organization analysis, HRA construction, or disease interpretation^{16,17}.

6.5 Future Directions

A major future direction is a unified multiscale biological foundation model. A useful multiscale system should connect nuclei, whole cells, FTUs, tissue compartments, and organ regions while preserving the relationships between these objects. This requires hierarchical training data, scale-aware prompts, and metrics that prevent one scale from dominating the evaluation. The goal is not simply one model for every mask, but one measurement framework in which objects at different biological scales remain organized with each other.

Interactive and human-in-the-loop segmentation is another important direction. Biologists and pathologists often recognize an incorrect mask quickly, but they need tools that make correction fast, traceable, and useful for model refinement. Promptable systems can combine points, boxes, scribbles, active learning, uncertainty estimates, and expert review in tools such as napari or whole-slide viewers. CellSeg1 points toward this direction by adapting SAM with LoRA from one image with only a few annotated cells²¹. The key challenge is to make model behavior predictable

enough that expert corrections improve the dataset and the deployed model rather than only the current image.

Multimodality and multiplexed imaging will require more flexible input representations. Future models should handle multiplex fluorescence, phase contrast, H&E, spatial transcriptomics imaging, mass cytometry imaging, and electron microscopy. These modalities differ in channel semantics, resolution, noise, and biological signal. Channel-aware encoders, modality-specific adapters, and parameter-efficient fine-tuning may extend segmentation systems without retraining every component from the beginning²².

Scalable whole-slide and 3D segmentation remains an open engineering and scientific problem. Whole-slide images require tiling, streaming inference, stitching, and global context aggregation. Volumetric microscopy requires memory-efficient 3D models and robust propagation across slices. Extending the methods in this thesis to these settings will require both model changes and suitable data infrastructure.

6.6 Final Perspective

The long-term objective of biological image segmentation is not to produce visually plausible masks. It is to create reliable computational representations of biological structure. This thesis contributes to that objective by connecting four levels of the problem. These are the data representation that defines the measured object, the metric that decides whether the object was recovered, the model prior or prompt that makes segmentation feasible, and the scale adaptation needed when the object changes from a cell to an FTU.

This perspective makes segmentation a bridge between microscopy and biological reasoning. When the bridge is weak, downstream analyses can become precise measurements of the wrong object. When it is strong, segmentation can support cell-state mapping, spatial neighborhoods, tissue-unit morphometry, atlas construction, and disease interpretation. The central lesson of the thesis is therefore simple but demanding: biological segmentation systems must be evaluated and designed as measurement systems for specific biological units. That standard is harder than predicting a mask, but it is the standard required for segmentation to become a trustworthy foundation for quantitative biology.

References

1. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28. arXiv: 1505.04597.
2. Kaiming He et al. “Mask R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2961–2969. DOI: 10.1109/iccv.2017.322. arXiv: 1703.06870.
3. Nicolas Carion et al. “End-to-End Object Detection with Transformers”. In: *Computer Vision – ECCV 2020*. Springer, 2020, pp. 213–229. DOI: 10.1007/978-3-030-58452-8_13. arXiv: 2005.12872.
4. Bowen Cheng et al. “Masked-Attention Mask Transformer for Universal Image Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1290–1299. DOI: 10.1109/cvpr52688.2022.00135. arXiv: 2112.01527.
5. Carsen Stringer et al. “Cellpose: a generalist algorithm for cellular segmentation”. In: *Nature Methods* 18.1 (Jan. 2021), pp. 100–106. DOI: 10.1038/s41592-020-01018-x.
6. Uwe Schmidt et al. “Cell Detection with Star-Convex Polygons”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2018*. Springer, 2018, pp. 265–273. DOI: 10.1007/978-3-030-00934-2_30. arXiv: 1806.03535.
7. Noah F. Greenwald et al. “Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning”. In: *Nature Biotechnology* 40.4 (Apr. 2022), pp. 555–565. DOI: 10.1038/s41587-021-01094-0.
8. Kevin J. Cutler et al. “Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation”. In: *Nature Methods* 19.11 (Nov. 2022), pp. 1438–1448. DOI: 10.1038/s41592-022-01639-4.
9. Alexander Kirillov et al. “Segment Anything”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4015–4026. arXiv: 2304.02643.
10. Nikhila Ravi et al. “SAM 2: Segment Anything in Images and Videos”. In: *International Conference on Learning Representations*. 2025. arXiv: 2408.00714.
11. Markus Marks et al. “CellSAM: a foundation model for cell segmentation”. In: *Nature Methods* 22.12 (Dec. 8, 2025), pp. 2585–2593. DOI: 10.1038/s41592-025-02879-w.

12. Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Springer, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48. arXiv: 1405.0312.
13. Annika Reinke et al. “Common Limitations of Image Processing Metrics: A Picture Story”. In: *arXiv preprint arXiv:2104.05642* (2022). arXiv: 2104.05642.
14. Taco Cohen and Max Welling. “Group Equivariant Convolutional Networks”. In: *Proceedings of the 33rd International Conference on Machine Learning*. PMLR, 2016, pp. 2990–2999. arXiv: 1602.07576.
15. Jasper Linmans et al. *Sample Efficient Semantic Segmentation Using Rotation Equivariant Convolutional Networks*. 2018. arXiv: 1807.00583.
16. Yashvardhan Jain et al. “Segmenting functional tissue units across human organs using community-driven development of generalizable machine learning algorithms”. In: *Nature Communications* 14.1 (Aug. 3, 2023), p. 4656. doi: 10.1038/s41467-023-40291-0.
17. Yashvardhan Jain et al. “Segmentation of human functional tissue units in support of a Human Reference Atlas”. In: *Communications Biology* 6.1 (July 19, 2023), p. 717. doi: 10.1038/s42003-023-04848-5.
18. Isaac Robinson et al. “RF-DETR: Neural Architecture Search for Real-Time Detection Transformers”. In: *International Conference on Learning Representations*. 2026. arXiv: 2511.09554.
19. Anwai Archit and Constantin Pape. “Revisiting foundation models for cell instance segmentation”. In: *Proceedings of The 9th International Conference on Medical Imaging with Deep Learning*. Vol. 315. Proceedings of Machine Learning Research. PMLR, 2026, pp. 3464–3495. arXiv: 2603.17845.
20. Anwai Archit et al. “Segment Anything for Microscopy”. In: *Nature Methods* 22.3 (Mar. 2025), pp. 579–591. doi: 10.1038/s41592-024-02580-4.
21. Peilin Zhou, Bo Du, and Yongchao Xu. “CellSeg1: Robust Cell Segmentation with One Training Image”. In: *arXiv preprint arXiv:2412.01410* (2024). doi: 10.48550/arxiv.2412.01410. arXiv: 2412.01410.
22. Jun Ma et al. “The multimodality cell segmentation challenge: toward universal solutions”. In: *Nature Methods* 21.6 (June 2024), pp. 1103–1113. doi: 10.1038/s41592-024-02233-6.

INDEX

Symbols

3D imaging, 60

A

Anchor DETR, 51

annotation quality, 62

annotation scheme, 62

automatic prompting, 50

average precision, 22, 67

B

benchmarking, 66

biological microscopy, 1

bounding-box prompt, 7, 48

BriFiSeg, 6, 52

C

cell morphology, 55

Cell Preprocessing and Evaluation

Workflow, 4, 14

cell segmentation, 7, 13, 48

CellFinder, 7, 48

Cellpose, 52

CellSAM, 7, 48

CellSAM-generalist, 56

CellSAM-specialist, 56

COCO AP, 64

COCO AP50, 67

COCO format, 3, 62

COCO JSON, 13

COCO mAP, 67

COCO metrics, 67

D

data conversion, 3

data leakage, 51

Data Science Bowl, 52

data split, 51

dataset construction, 50

dataset provenance, 4

DeepBacs, 6, 52

dense cellular image, 53

dense label mask, 3, 13

detector-guided prompting, 48

Dice coefficient, 22

distribution shift, 55

E

equivariance, 8

reflection, 8

rotation, 8

Equivariant Mask2Former, 7, 28

escnn, 28

evaluation

object-level, 21, 67

pixel-level, 22

F

F1 score, 22, 56

feature pyramid network, 35

few-shot learning, 56

finite rotation/reflection group, 8

foundation model, 48

FTU, *see* functional tissue unit

FTUFinder, 7, 87

FTURader, 7, 87

functional tissue unit, 1

G

ground-truth prompt, 53

group-equivariant convolution, 8

H

H&E staining, 7, 50
human-in-the-loop annotation, 61

I

inference, 51
instance segmentation, 4, 50
inter-annotator agreement, 56
intersection over union, 21, 64
 threshold, 67
invalid cell label, 62
IoU prediction head, 65

K

k-means clustering, 64

L

live-cell imaging, 57
LIVECell, 6, 51

M

Macenko normalization, 7, 96
mask decoder, 7, 48
mask encoding, 14
mask fidelity, 4
Mask2Former, 4
Mesmer, 53
model neck, 53

N

NMS, *see* non-maximum suppression
non-maximum suppression, 53
normalization
 intensity, 4

O

object detection, 51
 set prediction, 53
object matching, 21

Omnipose, 6, 52

overlapping objects, 53

P

Phase400, 6, 52
pixel-wise hashing, 62
point prompt, 50
polygon annotation, 3, 13
precision, 22, 67
precision-recall curve, 67
prompt encoder, 64
prompt engineering, 61
promptable segmentation, 4, 50
PyTorch, 65

R

recall, 22, 67
ResNet-50, 31
run-length encoding, 3, 13

S

SAM2, 4
Segment Anything Model, 4, 48
semantic segmentation, 4
small-object segmentation, 14
spatial biology, 57
spatial transcriptomics, 57
stain normalization, 7

T

thresholding, 64
tiling, 4
TissueNet, 6, 52
training
 two-stage, 55
training data distribution, 57

U

u-segment3D, 66

V

vision transformer, 35, 49

ViT, *see* vision transformer

W

whole-cell segmentation, 52

whole-slide image, 86

Y

YeastNet, 6, 52

YeaZ, 6

Z

zero-shot evaluation, 50