Quickly Tuning Foundation Models for Image Segmentation

Breenda Das[1] **Lennart Purucker**[1] **Timur Carstensen**[2,1] **Frank Hutter**[3,2,1]

[1]University of Freiburg
[2]ELLIS Institute Tübingen
[3]Prior Labs

Abstract Foundation models like SAM (Segment Anything Model) exhibit strong zero-shot image segmentation performance, but often fall short on domain-specific tasks. Fine-tuning these models typically requires significant manual effort and domain expertise. In this work, we introduce QTT-SEG, a meta-learning-driven approach for automating and accelerating the fine-tuning of SAM for image segmentation. Built on the Quick-Tune hyperparameter optimization framework, QTT-SEG predicts high-performing configurations using meta-learned cost and performance models, efficiently navigating a search space of over 200 million possibilities. We evaluate QTT-SEG on eight binary and five multiclass segmentation datasets under tight time constraints. Our results show that QTT-SEG consistently improves upon SAM's zero-shot performance and surpasses AutoGluon Multimodal, a strong AutoML baseline, on most binary tasks within three minutes. On multiclass datasets, QTT-SEG delivers consistent gains as well. These findings highlight the promise of meta-learning in automating model adaptation for specialized segmentation tasks. Code available at: Link.

1 Introduction

Pre-trained foundation models like SAM (Ravi et al., 2024) have revolutionized image segmentation by offering strong generalization across diverse domains. However, their performance often plateaus on specialized, domain-specific datasets where fine-tuning is necessary. Traditional adaptation methods typically require extensive manual tuning of hyperparameters and strategies (Hutter et al., 2015; Quinton et al., 2024; Sharma et al., 2019; Ogundokun et al., 2022), making them resource-intensive and difficult to scale.

Quick-Tune (Arango et al., 2023) addresses this bottleneck through meta-learned predictors that guide efficient hyperparameter selection, previously demonstrated for image classification and extended by Strangmann et al. (2024) to the language domain. In this work, we extend Quick-Tune to image segmentation and show that it can effectively adapt SAM across a wide range of domains with minimal manual effort. Evaluated on 13 benchmark segmentation datasets, Quick-Tune consistently improves over SAM's zero-shot performance and frequently outperforms AutoGluon — even under tight time budgets. These results underscore the potential of meta-learning to automate the adaptation of foundation models, enabling efficient and scalable performance optimization across diverse tasks and domains.

2 Background and Related Work

Quick-Tune. Quick-Tune (Arango et al., 2023) is a Bayesian Optimization (BO) framework for optimizing deep learning pipelines. It employs a probabilistic performance predictor $\hat{\ell}_\theta$ and a cost predictor $\hat{c}_w$ to select pipelines maximizing Multi-fidelity Expected Improvement while accounting for fine-tuning costs. Meta-learned parameters θ and w guide exploration of hyperparameter and model configurations. Quick-Tune uses deep-kernel GPs (Wistuba et al., 2022) and MLPs trained on historical data for performance and cost prediction. Initially applied to image classification, it has

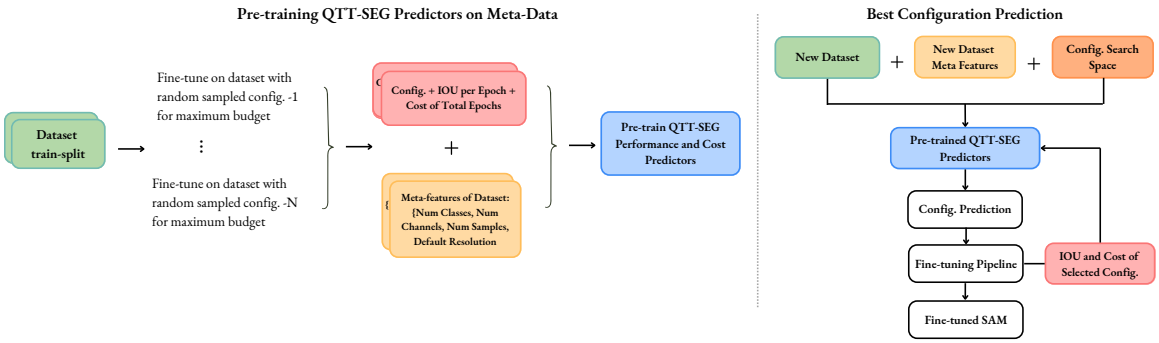
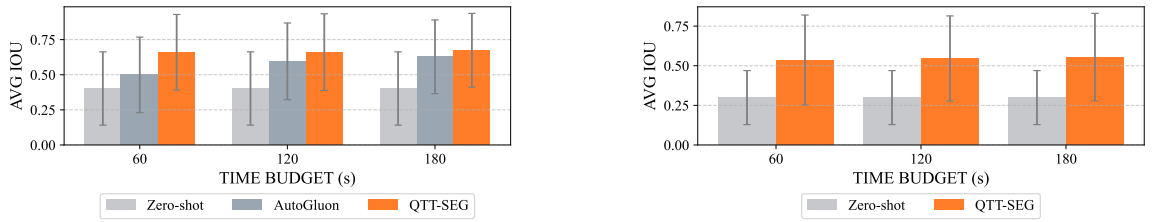


Figure 1: **Overview of QTT-SEG**: It first pre-trains performance and cost predictors using dataset meta-features, performance and cost traces from multiple configurations. These predictors are then used to guide efficient pipeline selection and fine-tuning on new datasets.



(a) Binary Segmentation

(b) Multiclass Segmentation

Figure 2: **Performance over Time Budgets**: Mean IoU (bars) and standard deviation (error bars) of Zero-shot, AG(Autogluon-Multimodal), and QTT-SEG across all binary and multiclass segmentation tasks. QTT-SEG shows consistent gains with longer budgets. *Note: AG is evaluated only on binary tasks due to missing out-of-the-box multiclass support.*

been extended to LLMs (Strangmann et al., 2024) and released as an open-source tool *Quick-Tune-Tool* (Rapant et al., 2024). Here, we extend it to semantic segmentation.

Fine-tuning SAM. Several methods adapt SAM (Ravi et al., 2024) using parameter-efficient tuning (e.g., SAM-PARSER (Peng et al., 2024)), generalization techniques (e.g., GSAM (Kato et al., 2024)), and few-shot strategies (Xie et al., 2024). These enhance cross-domain performance with limited labels. Our approach automates SAM tuning via meta-learning and BO, reducing manual effort while maintaining strong results.

AutoML Systems. Tools like AutoKeras (Jin et al., 2019), AutoPytorch (Zimmer et al., 2021), and NePS (Mallik et al., 2023) automate model and hyperparameter search. Others like AutoGluon MultiModal (Tang et al., 2024) and ZAP (Öztürk et al., 2022) focus on foundation model tuning. Unlike these, Quick-Tune leverages meta-learning of dataset meta-features and learning curves to guide search, enabling faster and more cost-efficient adaptation.

3 Quick-Tune-Tool for Semantic Segmentation (QTT-SEG)

Foundation model tuning for segmentation is difficult due to dataset diversity, resource constraints, and sensitivity to hyperparameters. QTT-SEG addresses this by efficiently tuning SAM (Ravi et al., 2024) through a structured BO pipeline (see Figure 1). It defines a rich hyperparameter space, trains performance predictors on learning curves, and evaluates robustness across datasets. We prompt SAM with bounding boxes extracted from ground truth masks, adding random perturbations to simulate noisy prompts.

Table 1: Binary Semantic Segmentation ranked by QTT-SEG gain over Zero-shot in 60s. Values represent mean accuracy with standard deviation subscripted.

Dataset	Zero-shot	60 SEC		120 SEC		180 SEC	
		AG	QTT-SEG	AG	QTT-SEG	AG	QTT-SEG
polyp	$0.495_{0.006}$	$0.328_{0.183}$	$\mathbf{0.850_{0.031}}$	$0.622_{0.027}$	$\mathbf{0.866_{0.008}}$	$0.711_{0.011}$	$\mathbf{0.867_{0.006}}$
lesion	$0.573_{0.028}$	$0.704_{0.046}$	$\mathbf{0.870_{0.008}}$	$0.802_{0.016}$	$\mathbf{0.863_{0.009}}$	$0.806_{0.013}$	$\mathbf{0.882_{0.008}}$
leaf	$0.377_{0.010}$	$0.501_{0.024}$	$\mathbf{0.657_{0.009}}$	$0.587_{0.032}$	$\mathbf{0.635_{0.034}}$	$0.633_{0.382}$	$\mathbf{0.646_{0.028}}$
covid	$0.342_{0.005}$	$0.397_{0.026}$	$\mathbf{0.612_{0.018}}$	$0.447_{0.029}$	$\mathbf{0.647_{0.013}}$	$0.499_{0.021}$	$\mathbf{0.636_{0.017}}$
eyes	$0.069_{0.002}$	$0.216_{0.102}$	$\mathbf{0.291_{0.028}}$	$\mathbf{0.323}_{0.025}$	$0.314_{0.023}$	$\mathbf{0.365_{0.013}}$	$0.323_{0.012}$
fiber	$0.007_{0.000}$	$0.203_{0.017}$	$\mathbf{0.229_{0.016}}$	$\mathbf{0.241_{0.007}}$	$0.196_{0.111}$	$0.249_{0.009}$	$\mathbf{0.250_{0.017}}$
cardiac	$0.764_{0.009}$	$0.733_{0.040}$	$\mathbf{0.875_{0.012}}$	$\mathbf{0.828_{0.009}}$	$0.866_{0.019}$	$0.835_{0.007}$	$\mathbf{0.886_{0.007}}$
chest	$0.591_{0.002}$	$\mathbf{0.909_{0.002}}$	$0.896_{0.010}$	$\mathbf{0.914_{0.004}}$	$0.902_{0.004}$	$\mathbf{0.919_{0.003}}$	$0.903_{0.014}$
Average	0.402	0.499	**0.660**	0.595	**0.661**	0.627	**0.674**

Table 2: Multiclass Semantic Segmentation ranked by QTT-SEG gain over Zero-shot in 60s. Values represent mean accuracy with standard deviation subscripted.

Dataset	Zero-shot	60 SEC (QTT-SEG)	120 SEC (QTT-SEG)	180 SEC (QTT-SEG)
US	$0.304_{0.002}$	$0.814_{0.012}$	$0.784_{0.088}$	$0.831_{0.013}$
human_parsing	$0.524_{0.006}$	$0.843_{0.006}$	$0.853_{0.008}$	$0.846_{0.014}$
golf	$0.151_{0.007}$	$0.362_{0.047}$	$0.383_{0.034}$	$0.391_{0.026}$
terrain	$0.399_{0.006}$	$0.466_{0.036}$	$0.494_{0.030}$	$0.483_{0.013}$
cholec	$0.117_{0.007}$	$0.196_{0.038}$	$0.216_{0.011}$	$0.221_{0.018}$
Average	0.299	0.536	0.546	0.554

Search Space: We designed an extensive search space covering fine-tuning, optimization, and augmentation hyperparameters to maximize SAM's performance on new datasets. The search space spans over 200 million configurations. Key groups include LoRA (Hu et al., 2022) application, data augmentation, optimizer, loss, learning rates, schedulers, and warm-up phases (see Table 3).

Meta Training: We sampled 2 000 configuration–dataset pairs across all selected binary and multiclass segmentation datasets. Each sampled configuration was fine-tuned on its corresponding dataset for 10 epochs, while tracking performance and training cost. This resulted in a meta-dataset of configuration–performance and cost pairs, which were then used to pre-train predictors.

4 Experiments and Results

Benchmark Setup. We evaluate QTT-SEG on 8 binary and 5 multiclass semantic segmentation datasets that we manually curated (details in Appendix B).

Experimental Setup. Each experiment uses 128 configurations, tuned under time budgets of 60, 120, and 180 seconds. We subsample 100 images and masks per dataset using five random seeds. Tuning is performed independently for each seed using QTT-SEG, reporting mean IoU and standard deviation. For each run, we exclude the target dataset's learning curves from metadata, ensuring the predictors are trained only on other datasets, hence enabling generalization to unseen data.

For binary segmentation tasks, we compare QTT-SEG against both AutoGluon Multimodal (Tang et al., 2024) and SAM's zero-shot performance, using consistent time budgets and random seeds. For multiclass tasks, only SAM zero-shot is used as a baseline, as AutoGluon's support for multiclass segmentation is not clearly documented. We ran all experiments on a single NVIDIA GeForce RTX 2080 Ti GPU with 11 GB RAM.

Results. Figure 2 compares average performance of QTT-SEG on all datasets with baselines. Table Table 1 shows QTT-SEG's performance versus AutoGluon and zero-shot SAM on binary datasets and Table Table 2 covers multiclass datasets. (i) **QTT-SEG consistently outperforms zero-shot baselines**

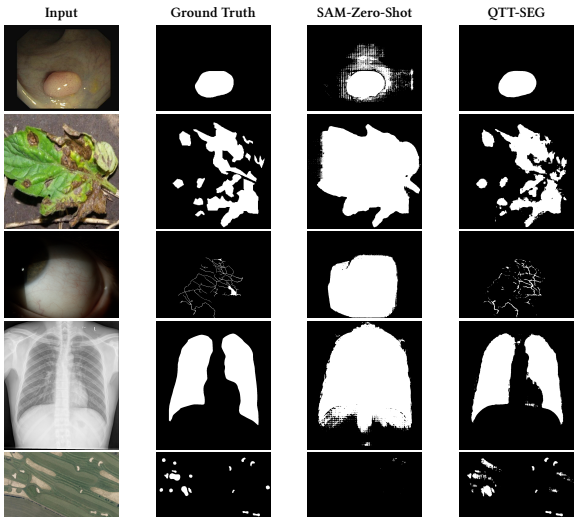


Figure 3: **Comparison of segmentation results across five datasets**: polyp, leaf, eyes, chest, golf at 60 seconds. The figure highlights the performance consistency of QTT-SEG across diverse domains. All labels are white and background is black.

across all datasets. On average, QTT-SEG achieves a 76.47% improvement in IoU over zero-shot baselines across all datasets at 180 seconds. (ii) **QTT-SEG is a compute-efficient alternative**. QTT-SEG reaches 97.3% of its final accuracy within 60 seconds and maintains better accuracy than AutoGluon on 6 out of 8 binary classification datasets at the 180-second mark. This highlights QTT-SEG's efficiency compared to AutoGluon's incremental but slower convergence, likely due to QTT-SEG's meta-trained predictors enabling faster convergence to superior configurations, while AutoGluon relies on a more exhaustive search. (iii) **QTT-SEG is robust to local optima**. The relative gain of 1.97% between 120 and 180 seconds, compared to a smaller 0.95% gain between 60 and 120 seconds, (avg. on binary) suggests that QTT-SEG continues to improve beyond early plateaus. While relative gains alone do not prove escaping local optima, this pattern indicates effective refinement of configurations and is further supported by superior performance on 6 out of 8 datasets. (iv) **QTT-SEG demonstrates stronger domain generalization**. Although AutoGluon performs better on datasets like 'chest' and 'eyes', QTT-SEG outperforms it on the others. Figure 3 shows representative results where each framework excels. With a higher average improvement over zero-shot (67.16% vs. 55.72% at 180 seconds), QTT-SEG appears more robust to domain shifts overall. See Appendix C for sample predictions on all datasets. (v) **QTT-SEG scales effectively to multiclass datasets**. On multiclass segmentation tasks, QTT-SEG surpasses zero-shot baselines by an average of 85.28% at 180 seconds. (vi) **QTT-SEG demonstrates stability across folds**. QTT-SEG proves to be a highly reliable tuning framework, as indicated by its low standard deviation across folds within the same dataset.

Conclusion. QTT-SEG is an efficient AutoML framework that consistently outperforms zero-shot baselines and converges faster than AutoGluon across diverse segmentation tasks. Its meta-trained predictors enable rapid tuning and robust escape from local optima, leading to superior accuracy and stability. While AutoGluon excels on certain modalities, QTT-SEG shows greater overall robustness to domain shifts. Future work would focus on expanding the search space and incorporating richer domain knowledge to further improve performance.

Acknowledgements. L.P. acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under SFB 1597 (SmallData), grant number 499552394. F.H. acknowledges the financial support of the Hector Foundation.

References

Arango, S. P., Ferreira, F., Kadra, A., Hutter, F., and Grabocka, J. (2023). Quick-tune: Quickly learning which pretrained model to finetune and how. *arXiv preprint arXiv:2306.03828*.

Degerli, A., Kiranyaz, S., Chowdhury, M. E., and Gabbouj, M. (2022). Osegnet: Operational segmentation network for covid-19 detection using chest x-ray images. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2306–2310. IEEE.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2022). Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Hutter, F., Lücke, J., and Schmidt-Thieme, L. (2015). Beyond manual tuning of hyperparameters. *KI-Künstliche Intelligenz*, 29:329–337.

Jin, H., Song, Q., and Hu, X. (2019). Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1946–1956.

Kato, S., Mitsuoka, H., and Hotta, K. (2024). Generalized sam: Efficient fine-tuning of sam for variable input image sizes. *arXiv preprint arXiv:2408.12406*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Mallik, N., Bergman, E., Hvarfner, C., Stoll, D., Janowski, M., Lindauer, M., Nardi, L., and Hutter, F. (2023). Priorband: Practical hyperparameter optimization in the age of deep learning. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS 2023)*.

Ogundokun, R. O., Maskeliūnas, R., and Damaševičius, R. (2022). Human posture detection using image augmentation and hyperparameter-optimized transfer learning algorithms. *Applied Sciences*, 12(19):10156.

Öztürk, E., Ferreira, F., Jomaa, H., Schmidt-Thieme, L., Grabocka, J., and Hutter, F. (2022). Zero-shot automl with pretrained models. In *International Conference on Machine Learning*, pages 17138–17155. PMLR.

Paszke, A. (2019). Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Peng, Z., Xu, Z., Zeng, Z., Yang, X., and Shen, W. (2024). Sam-parser: fine-tuning sam efficiently by parameter space reconstruction. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Quinton, F., Presles, B., Leclerc, S., Nodari, G., Lopez, O., Chevallier, O., Pellegrinelli, J., Vrigneaud, J.-M., Popoff, R., Meriaudeau, F., et al. (2024). Navigating the nuances: comparative analysis and hyperparameter optimisation of neural architectures on contrast-enhanced mri for liver and liver tumour segmentation. *Scientific Reports*, 14(1):3522.

Rapant, I., Purucker, L., Ferreira, F., Arango, S. P., Kadra, A., Grabocka, J., and Hutter, F. (2024). Quick-tune-tool: A practical tool and its user guide for automatically finetuning pretrained models. In *AutoML Conference 2024 (Workshop Track)*.

Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., et al. (2024). Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*.

Sharma, A., van Rijn, J. N., Hutter, F., and Müller, A. (2019). Hyperparameter importance for image classification by residual neural networks. In *Discovery Science: 22nd International Conference, DS 2019, Split, Croatia, October 28–30, 2019, Proceedings 22*, pages 112–126. Springer.

Smith, L. N. and Topin, N. (2019). Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE.

Strangmann, T., Purucker, L., Franke, J. K., Rapant, I., Ferreira, F., and Hutter, F. (2024). Transfer learning for finetuning large language models. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*.

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., and Jorge Cardoso, M. (2017). Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*, pages 240–248. Springer.

Tang, Z., Fang, H., Zhou, S., Yang, T., Zhong, Z., Hu, T., Kirchhoff, K., and Karypis, G. (2024). Autogluon-multimodal (automm): Supercharging multimodal automl with foundation models. *arXiv preprint arXiv:2404.16233*.

Wagner, F. (2023). Fiber segmentation dataset.

Wistuba, M., Kadra, A., and Grabocka, J. (2022). Supervising the multi-fidelity race of hyperparameter configurations. *Advances in Neural Information Processing Systems*, 35:13470–13484.

Xie, B., Tang, H., Duan, B., Cai, D., and Yan, Y. (2024). Masksam: Towards auto-prompt sam with mask classification for medical image segmentation. *arXiv preprint arXiv:2403.14103*.

Zimmer, L., Lindauer, M., and Hutter, F. (2021). Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE transactions on pattern analysis and machine intelligence*, 43(9):3079–3090.

5 [Optional] Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes]

   (c) Did you discuss any potential negative societal impacts of your work? [N/A]

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? (see `https://2022.automl.cc/ethics-accessibility/`) [Yes]

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources, etc.)? [Yes]

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning details and results, etc.)? [Yes]

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes]

   (d) Did you report the uncertainty of your results (e.g., the standard error across random seeds or splits)? [Yes]

   (e) Did you report the statistical significance of your results? [N/A]

   (f) Did you use enough repetitions, datasets, and/or benchmarks to support your claims? [Yes]

   (g) Did you compare performance over time and describe how you selected the maximum runtime? [Yes]

   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

   (i) Did you run ablation studies to assess the impact of different components of your approach? [Yes]

3. With respect to the code used to obtain your results...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all dependencies (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation instructions, and execution commands (either in the supplemental material or as a URL)? [Yes]

   (b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes]

   (c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes]

   (d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes]

   (e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes]

4. If you used existing assets (e.g., code, data, models)...

(a) Did you citep the creators of used assets? [Yes]

(b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [N/A]

(c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you created/released new assets (e.g., code, data, models)...

(a) Did you mention the license of the new assets (e.g., as part of your code submission)? [Yes]

(b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes]

6. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to institutional review board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

7. If you included theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [N/A]

(b) Did you include complete proofs of all theoretical results? [N/A]

Table 3: **QTT-SEG Search Space**: Key hyperparameters for segmentation tuning.

Hyper-Parameters	**Choices**
LoRA Application	Image Encoder Attention & MLP Layers
LoRA Enabled	0, 1
LoRA Rank	4, 8, 16
LoRA Dropout	0.0, 0.1
Optimizer	AdamW
Weight Decay	0.0, 1e-5, 5e-5, 1e-4
Learning Rate	1e-5, 1.2e-5, 1.5e-5, 2e-5, 2.5e-5, 3.5e-5, 5e-5, 6e-5, 6.5e-5, 0.0001, 0.00012, 0.00018, 0.00025, 0.00032, 0.0004, 0.00048, 0.0005, 0.00055, 0.0008, 0.001, 0.0015, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007
Loss Function	BCE + Dice
Data Augmentation	Horizontal Flip (0,1), Vertical Flip (0,1), Random Rotate (0,1)
Learning Rate Schedulers	Cosine, OneCycle, Plateau, Cosine_Warm, Step, Poly
Plateau Parameters:	factor: 0.1, 0.5, 0.8 patience: 0, 1, 2
Cosine_Warm Parameters:	T_0: 2, 3, 5 T_{mult}: 1, 2
OneCycle Parameters:	pct_start: 0.030–0.100 (step 0.005) div_factor: 10–100 final_div_factor: 10–1000
Step Scheduler Parameters:	step_size: 3, 5
Polynomial Scheduler Parameters:	power: 0.5, 0.9, 1.0

A Search-Space

The QTT-SEG search space is designed to enable flexible and effective adaptation of large segmentation models across diverse domains and computational budgets. It includes a range of architectural and training hyperparameters, with a focus on efficiency and robustness. Core components of the space involve the optional use of LoRA (Low-Rank Adaptation), applied to the attention and MLP layers of the image encoder, with tunable settings for activation, rank, and dropout. Optimization is performed using the AdamW optimizer (Kingma and Ba, 2014), with a finely grained range of learning rates spanning from 1e-5 to 7e-3, sampled in a rough-logarithmic progression to allow fine control in both low and high learning rate regimes. Data augmentation strategies such as horizontal and vertical flips, and random rotations are included as binary choices. The loss function combines Binary Cross-Entropy (BCE) with Dice loss (Sudre et al., 2017) to handle class imbalance common in segmentation tasks. A variety of learning rate schedulers are supported: Cosine (Loshchilov and Hutter, 2016), OneCycle (Smith and Topin, 2019), Plateau, Step and Polynomial (Paszke, 2019), each with its own tunable parameters to better adapt to dataset and time-budget characteristics. This rich and modular search space, detailed in Table Table 3, allows QTT-SEG to discover effective configurations in a time-constrained tuning setting.

B Datasets

We present a brief overview and sources of each dataset used to benchmark QTT-SEG's performance. Our selection prioritizes diversity across domains and real-world relevance in semantic segmentation tasks, while ensuring all datasets are openly accessible from reputable platforms such as Kaggle, Hugging Face, and DatasetNinja.

Binary. We tested on eight binary classification datasets:

- **polyp**: This dataset comprises 612 colonoscopy frames extracted from 29 video sequences, each annotated with a binary mask highlighting polyps. It is designed for semantic segmentation tasks in medical imaging, particularly for polyp detection in colonoscopy videos. The dataset is valuable for training and evaluating models in early colorectal cancer detection. Source: CVC-ClinicDB on Kaggle.
- **lesion**: This dataset comprises 900 dermoscopic images of skin lesions, each paired with a corresponding binary segmentation mask. The images are provided in JPEG format, while the masks are in PNG format, with pixel values of 0 for background and 255 for the lesion area. This dataset was used in the ISIC 2016 Challenge for automated skin lesion segmentation. Source: ISIC 2016 Task 1 Training Data on Kaggle.
- **leaf**: This dataset comprises 588 images of diseased leaves, each paired with a corresponding segmentation mask. The data collection is based on the PlantDoc images, making it suitable for training and evaluating semantic segmentation models in plant disease detection tasks. Source: Leaf Disease Segmentation Dataset on Kaggle.
- **covid** (Degerli et al., 2022): This extensive dataset comprises over 121,000 chest X-ray images, including 9,258 annotated COVID-19 cases, sourced from Qatar University, Tampere University, and Hamad Medical Corporation. It supports both classification and segmentation tasks, featuring ground-truth masks for lesion areas. The dataset is valuable for training and evaluating deep learning models in medical image analysis. Source: QaTa-COV19 on Kaggle.
- **eyes**: This dataset comprises 962 retinal images annotated for instance segmentation, focusing on microvascular structures. It includes 4,277 labeled vessels and 313 unlabeled images, with a 2:1 train-test split. The dataset is suitable for training and evaluating models in medical image segmentation tasks. Source: Eyes Microcirculation on Dataset Ninja.
- **fiber** (Wagner, 2023): This dataset consists of 3 spatially disjoint volumes, each with dimensions $20 \times 512 \times 512$ voxels (voxel size: 4 μm). It is designed for fiber segmentation tasks in CT scans of concrete, providing a valuable resource for training and evaluating segmentation models in material science applications. Source: PE-Fibers on Kaggle.
- **cardiac**: This dataset comprises X-ray images of cardiac catheterization procedures, each with dimensions of 512x512 pixels in PNG format. The images are designed for training and evaluating models in medical image analysis tasks, particularly in the context of cardiovascular imaging. Source: Cardiac Catheterization X-Ray PNG 512x512 on Kaggle.
- **chest**: This dataset contains chest X-ray images with corresponding lung masks, designed for lung segmentation tasks. It includes both original and pre-processed grayscale images, along with binary masks for the left and right lungs. The dataset is suitable for training and evaluating medical image segmentation models. Source: Chest X-ray Lung Segmentation on Kaggle.

Multi-class. We tested on five multiclass datasets:

- **US**: This dataset comprises 617 real abdominal ultrasound scans, each annotated with manual segmentations of several abdominal organs, including the liver, kidneys, gallbladder, and spleen. The dataset is designed for training and evaluating models in medical image segmentation tasks, particularly in the context of abdominal ultrasound imaging. Source: USSimAndSegm on Kaggle.
- **human_parsing**: This dataset contains 17,706 images with corresponding segmentation masks, sourced from the ATR dataset. It is designed for human parsing tasks, providing detailed pixel-level annotations of clothing and body parts. The dataset is useful for training and evaluating human segmentation models. Source: Human Parsing Dataset on Hugging Face.

- **golf**: This dataset comprises 1,123 RGB orthophotos collected from 107 Danish golf courses during the spring season. The images are annotated for instance segmentation tasks, focusing on classes such as greens, fairways, tees, bunkers, and water hazards. An additional 108 images are provided for testing, with manual masking of non-course areas to facilitate course rating computations. The dataset is valuable for training and evaluating models in sports and geospatial analysis. Source: Danish Golf Courses Orthophotos on Kaggle.
- **terrain**: This dataset comprises 1,000 synthetic images generated using the Unity engine, each annotated with pixel-level semantic labels. It is designed for training and evaluating semantic segmentation models in mobile robotics applications. The dataset includes various terrain types such as grass, dirt, and pavement, providing a diverse set of scenes for model training. Source: Vale Semantic Terrain Segmentation on Kaggle.
- **cholec**: This dataset is a semantic segmentation benchmark derived from the Cholec80 dataset. It consists of 8,080 annotated frames from 17 laparoscopic cholecystectomy videos, labeled for 13 different classes including surgical instruments and anatomical structures. The dataset is useful for evaluating real-time segmentation models in surgical scenes. Source: CholecSeg8k on Kaggle.

C Sample Prediction masks for all datasets

We present sample prediction masks for all datasets by QTT-SEG and compare with zero-shot baselines.

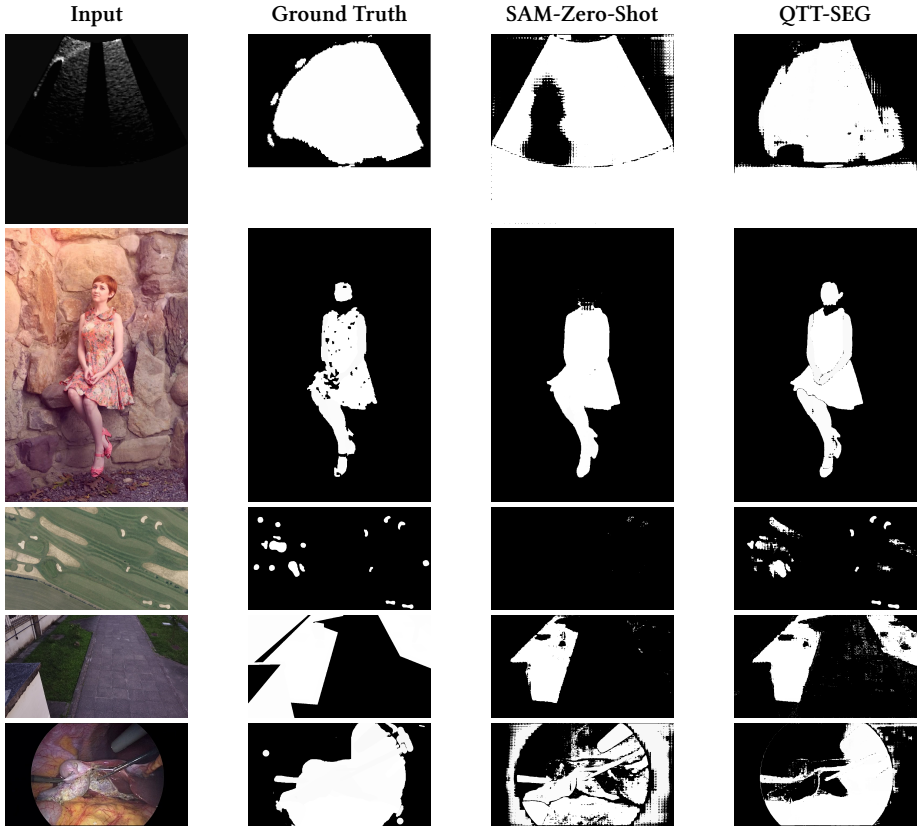


Figure 4: **Comparison of segmentation results across Multiclass datasets**: QTT-SEG demonstrates efficient domain generalization under 60 seconds of tuning. All labels are white and the background is black. Order top to bottom: US, human_parsing, golf, terrain, cholec.

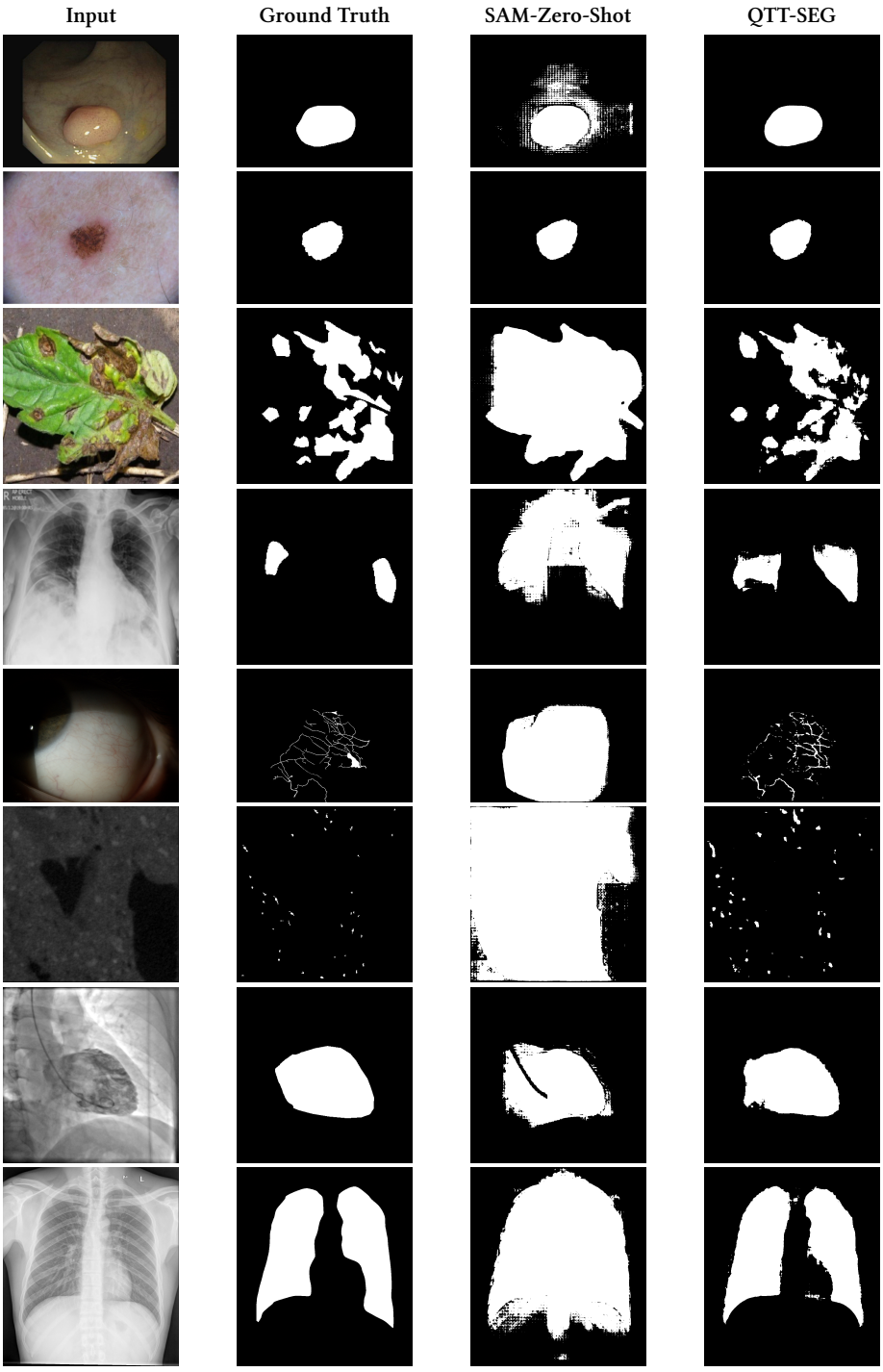


Figure 5: **Comparison of segmentation results across binary datasets**: QTT-SEG demonstrates efficient domain generalization under 60 seconds of tuning. All labels are white and the background is black. Order from top to bottom: polyp, lesion, leaf, covid, eyes, fiber, cardiac, chest.