Your ViT is Secretly an Image Segmentation Model

Supplementary Material

Appendix

Table of contents:

A. Implementation Details

A.1. Models

Visualizations of model configurations. In Sec. 3.2, we explain how we gradually remove task-specific components. We visualize the architectures of the resulting intermediate configurations in Fig. A. Here, the subscript $\boldsymbol{F}_i$ indicates that the features have a resolution of $\frac{1}{i}$ of the input image. The visualized model numbers correspond to those reported in Tab. 1.

Libraries. For Mask2Former [5], we use the implementation of `Huggingface Transformers` [23]. For pre-trained models, we use `timm` [22].

Pre-trained models. In Tab. A, we specify the `timm` model weights that we use for the experiments in this work. To support a patch size of 16×16 and different input sizes, we resize the patch embedding kernel and positional embeddings of pre-trained models following the FlexiViT [2] implementation of `timm`. Specifically, the patch embedding kernel is resized to a 16×16 patch size by approximately inverting the effect of patch resizing. The positional embeddings are resized to the required token grid size by using bicubic interpolation. The patch embedding kernel and positional embeddings are resized prior to fine-tuning, and keep the same size during fine-tuning.

Queries. In accordance with Mask2Former [5], the models for panoptic and instance segmentation use $K = 200$ queries, while the models for semantic segmentation use $K = 100$ queries. For ViT-S and ViT-B we use $L_2 = 3$, for ViT-L we use $L_2 = 4$, and for ViT-g we use $L_2 = 5$. For EoMT, adding 200 tokens to a model that processes 640×640 images with a 16×16 patch size results in an increase of 12.5% of the tokens processed by a ViT block, *but only for the last L_2 ViT blocks*. As $L_1 = 20$ and $L_2 = 4$ for ViT-L, the total number of tokens processed in the entire ViT increases by only 2.1%.

A.2. Training

Augmentation. During training, we apply the same data augmentation techniques as used by Mask2Former [5]. Specifically, training images undergo random horizontal flipping, random scale jittering, padding if necessary, and random cropping. Random color jittering is additionally applied for ADE20K [24] and Cityscapes [6]. For panoptic and instance segmentation, we use large-scale jitter [11] (between $0.1\times$ and $2.0\times$), and for semantic segmentation

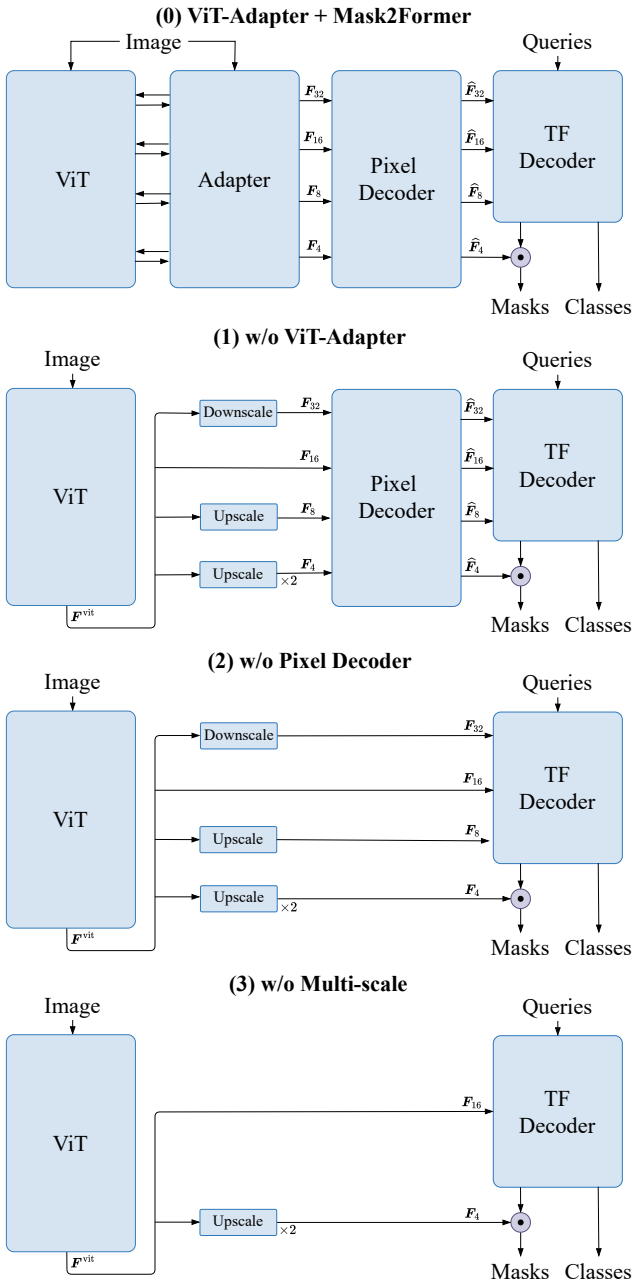


Figure A. **Removing task-specific components.** We visualize the architectures of the resulting intermediate configurations.

Model	Pre-training	`timm` model
ViT-g	DINOv2 [7, 19]	`vit_giant_patch14_reg4_dinov2`
ViT-L	DINOv2 [7, 19]	`vit_large_patch14_reg4_dinov2`
ViT-B	DINOv2 [7, 19]	`vit_base_patch14_reg4_dinov2`
ViT-S	DINOv2 [7, 19]	`vit_small_patch14_reg4_dinov2`
ViT-L	EVA-02 [10]	`eva02_large_patch14_224.mim_m38m`
ViT-L	DeiT-III (ImageNet-21K) [8, 20]	`deit3_large_patch16_384.fb_in22k_ft_in1k`
ViT-L	DeiT-III (ImageNet-1K) [8, 20]	`deit3_large_patch16_384.fb_in1k`

Table A. **Model specification.** For each ViT backbone [9] used in this work, we specify the `timm` model [22] that we use.

	Method	Params	GFLOPs	FPS	Panoptic Quality (PQ)			Average Precision (AP)			
					All	Things	Stuff	All	Large	Medium	Small
(0)	ViT-Adapter + Mask2Former	349M	830	29	57.1	62.7	48.7	47.6	73.2	53.4	23.4
(1)	↳ w/o ViT-Adapter	342M	700	36	56.7	62.3	48.3	46.9	72.7	52.9	22.7
(2)	↳ w/o Pixel decoder	337M	685	62	56.9	62.3	48.6	46.8	73.1	52.6	22.1
(3)	↳ w/o Multi-scale	328M	673	64	56.7	62.2	48.4	46.2	73.1	52.3	21.4
(4)	↳ w/o Transformer decoder	316M	828	61	56.2	61.4	48.4	45.6	72.1	51.4	20.8
(5)	↳ w/o Masking = **EoMT**	316M	669	128	56.0	61.2	48.2	45.2	72.2	51.0	20.3

Table B. **From ViT-Adapter + Mask2Former to EoMT in detail.** Evaluated on COCO *val2017*.

we use normal-scale jitter (between $0.5\times$ and $2.0\times$).

Loss function. To supervise our models, we adopt the same loss function as Mask2Former [5]. Specifically, across all tasks and datasets, we use the cross-entropy (CE) loss for the class logits, and the binary-cross entropy (BCE) and the Dice loss [17] for the mask logits. The individual losses are weighted using scalars, resulting in the total loss function:

$$\mathcal{L}_{\text{tot}} = \lambda_{\text{bce}}\mathcal{L}_{\text{bce}} + \lambda_{\text{dice}}\mathcal{L}_{\text{dice}} + \lambda_{\text{ce}}\mathcal{L}_{\text{ce}}, \quad (1)$$

where λ_{bce}, λ_{dice}, and λ_{ce} are set to 5.0, 5.0, and 2.0, respectively, following Mask2Former [5].

Learning rate warm-up. We use a two-stage linear learning rate warm-up for all models. In practice, we first warm-up the randomly initialized parameters for 500 iterations, while keeping the pre-trained parameters frozen. After 500 iterations, we warm-up the pre-trained parameters for 1000 iterations. In both cases, the initial learning rate is set to 0.

A.3. Evaluation

Image processing. For panoptic and instance segmentation, we use padded inference, resizing the longer side of the image to the input size, and padding the shorter side with zeros to create a square image. For semantic segmentation, we apply windowed inference, resizing the shorter side of the image to the input size, and processing the image through the model in several proportionally spaced square crops, in a sliding-window manner [14].

Efficiency measurements. For existing works, we report FLOPs from the respective papers but measure FPS the same way that we measure it for our models, on the same hardware. For ViT-Adapter + M2F and our models, we calculate the FLOPs ourselves. When measuring FPS, `torch.compile` [1] is disabled for Mask2Former [5] with Swin-L [16] on ADE20K [24] due to compilation errors. On COCO [15], `torch.compile` only yields a small speedup for this model ($< 10\%$). Additionally, mixed precision is not supported for OneFormer [13] with DiNAT-L [12], thus we use full precision here.

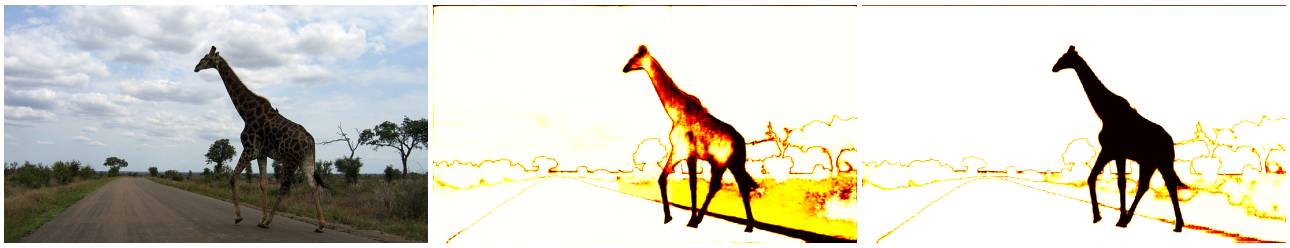
Token merging. For our token merging experiment in Sec. 4.3 and Tab. 9, we evaluate the throughput of the model in *images per second*, following existing work for token merging [3, 18]. This means that we use a batch size of 32, apply ALGM [18] for token merging, and report the number of images that are processed per second, averaged over the entire validation set. ALGM adaptively determines the number of tokens that should be merged per image, based on image complexity. To allow batch processing, we identify the lowest number of mergeable tokens per image across the batch according to the ALGM token merging criterion, and use that number of merged tokens for all images in the batch.

Importantly, ALGM is applied only during inference. Thus, the throughput improvement in Tab. 9 is achieved simply by applying ALGM to EoMT and processing batches of images, with no additional training required.

B. Detailed Experimental Analysis

From ViT-Adapter + M2F to EoMT in detail. In Tab. B, we provide more detailed results on the impact of the removal of task-specific components on both panoptic and instance segmentation on COCO [15]. For panoptic segmentation, we not only report the overall Panoptic Quality (PQ), but also separately the PQ for countable thing classes (PQ^{th}) and uncountable stuff classes (PQ^{st}). Similarly, for instance segmentation, we separately report AP for large (AP^{L}), medium (AP^{M}), and small objects (AP^{S}).

General applicability of mask annealing. In Tab. C, we assess the effect of our mask annealing strategy for both EoMT and the ViT-Adapter + M2F baseline. The results



(1) Input image (2) ViT-Adapter + M2F [4, 5] confidence (3) **EoMT** (Ours) confidence

Figure B. **Qualitative comparison of out-of-distribution (OOD) confidence estimation.** EoMT reliably assigns low confidence to the full OOD object, while ViT-Adapter + M2F only does so partially. Darker colors indicate lower confidence. Trained on Cityscapes *train* [6], evaluated on BRAVO [21].

Training	Inference	Panoptic Quality (PQ)	
		EoMT	ViT-Ad. + M2F
✓ Masking	✓ Masking	56.2	57.1
✗ w/o Masking	✗ w/o Masking	53.2$^{\downarrow 3.0}$	54.0$^{\downarrow 3.1}$
✓→✗ *Mask annealing*	✗ w/o Masking	56.0$^{\downarrow 0.2}$	56.8$^{\downarrow 0.3}$

Table C. ***Mask annealing***. Effective for both EoMT and ViT-Adapter + M2F [4, 5]. When never masking, intermediate masks are not predicted or supervised. Evaluated on COCO *val2017* [15].

# Blocks (L_2)	Params	GFLOPs	FPS	PQ
9	316	688	126	55.7
6	316	676	127	55.7
4	316	669	128	**56.0**
2	316	660	128	55.4

Table D. **Number of blocks that process queries.** The model with $L_2 = 4$ achieves the best PQ, while FPS is not significantly affected by changing L_2. Evaluated on COCO *val2017* [15].

demonstrate the general applicability of mask annealing, as it is also effective for ViT-Adapter + M2F.

Number of blocks that process queries. In Tab. D, we examine the impact of varying L_2, *i.e.*, the number of ViT blocks in EoMT that process queries as well as patch tokens. EoMT demonstrates stable performance across different configurations, with the highest PQ for ViT-L observed around $L_2 = 4$, while the prediction speed in FPS is not significantly affected by changing L_2. Consequently, we set $L_2 = 4$ as the default configuration for ViT-L.

C. Out-of-distribution Confidence Estimation

In Sec. 4.3, we discuss the out-of-distribution (OOD) generalization capabilities of EoMT. There, we show that DINOv2-based models, such as EoMT, significantly outperform non-ViT-based models such as Swin [16] in OOD generalization despite similar in-distribution (ID) performance.

Next, we also assess how well different models distinguish OOD regions from ID regions with their confidence scores. OOD regions, as defined in the BRAVO [21] benchmark, refer to novel object classes that were not present in the training data. We report the $\text{AUPRC}_{\text{OOD}}$ metric, which quantifies the model's ability to assign lower confidence to these unseen objects, ensuring they can be correctly identified as OOD.

Method	Backbone	Pre-training	$\text{AUPRC}_{\text{OOD}}$
M2F [5]	Swin-L [16]	IN21K	56.8
M2F‡ [5]	ViT-Adapter-L‡ [4]	DINOv2	68.7
EoMT (Ours)	ViT-L [9]	DINOv2	**89.7**

Table E. **Quantitative comparison of out-of-distribution (OOD) confidence estimation.** EoMT achieves the highest $\text{AUPRC}_{\text{OOD}}$, demonstrating its superior confidence estimation. Trained on Cityscapes *train* [6], evaluated on BRAVO [21]. ‡Our re-implementation.

As shown in Tab. E, EoMT achieves an $\text{AUPRC}_{\text{OOD}}$ of 89.7, significantly outperforming ViT-Adapter + M2F [4, 5] with a score of 68.7 and Swin [16] + M2F with a score of 56.8. The visualization in Fig. B further highlights that EoMT consistently assigns low confidence to the OOD object while maintaining high confidence for ID regions. In contrast, ViT-Adapter + M2F [4, 5] fails to reliably assign low confidence to all OOD pixels.

D. Qualitative Examples

In Fig. C we visualize predictions of ViT-Adapter + M2F [4, 5] and EoMT for panoptic segmentation on COCO [15].

(1) Input images

(2) Ground-truth annotations

(3) ViT-Adapter + M2F [4, 5] predictions

(4) **EoMT** (Ours) predictions

Figure C. **Qualitative examples for panoptic segmentation on COCO [15].** Using DINOv2-g [19] and a 1280 × 1280 input size.

References

[1] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, et al. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *ASPLOS*, 2024. 2

[2] Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. FlexiViT: One Model for All Patch Sizes. In *CVPR*, 2023. 1

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token Merging: Your ViT But Faster. In *ICLR*, 2023. 2

[4] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision Transformer Adapter for Dense Predictions. In *ICLR*, 2023. 3, 4

[5] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. In *CVPR*, 2022. 1, 2, 3, 4

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*, 2016. 1, 3

[7] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision Transformers Need Registers. In *ICLR*, 2024. 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. 2, 3

[10] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. EVA-02: A Visual Representation for Neon Genesis. *Image and Vision Computing*, 2024. 2

[11] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation. In *CVPR*, 2021. 1

[12] Ali Hassani and Humphrey Shi. Dilated Neighborhood Attention Transformer. *arXiv preprint arXiv:2209.15001*, 2022. 2

[13] Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One Transformer To Rule Universal Image Segmentation. In *CVPR*, 2023. 2

[14] Tommie Kerssies, Daan De Geus, and Gijs Dubbelman. How to Benchmark Vision Foundation Models for Semantic Segmentation? In *CVPR Workshops*, 2024. 2

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014. 2, 3, 4

[16] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. In *ICCV*, 2021. 2, 3

[17] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *3DV*, 2016. 2

[18] Narges Norouzi, Svetlana Orlova, Daan de Geus, and Gijs Dubbelman. ALGM: Adaptive Local-then-Global Token Merging for Efficient Semantic Segmentation with Plain Vision Transformers. In *CVPR*, 2024. 2

[19] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning Robust Visual Features without Supervision. *TMLR*, 2024. 2, 4

[20] Hugo Touvron, Matthieu Cord, and Hervé Jégou. DeiT III: Revenge of the ViT. In *ECCV*, 2022. 2

[21] Tuan-Hung Vu, Eduardo Valle, Andrei Bursuc, Tommie Kerssies, Daan de Geus, Gijs Dubbelman, Long Qian, Bingke Zhu, Yingying Chen, Ming Tang, Jinqiao Wang, Tomáš Vojíř, Jan Šochman, Jiří Matas, Michael Smith, Frank Ferrie, Shamik Basu, Christos Sakaridis, and Luc Van Gool. The BRAVO Semantic Segmentation Challenge Results in UNCV2024. In *ECCV Workshops*, 2024. 3

[22] Ross Wightman. PyTorch Image Models, 2019. 1, 2

[23] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP Demos*, 2020. 1

[24] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene Parsing through ADE20K Dataset. In *CVPR*, 2017. 1, 2