VidEoMT: Your ViT is Secretly Also a Video Segmentation Model

Supplementary Material

Appendix

Table of contents

A. Implementation Details

A.1. Training

Following state-of-the-art models CAVIS [9], DVIS-DAQ [21] and DVIS++ [20], we adopt a DINOv2-pretrained ViT [6, 16] as the backbone of VidEoMT, and we train our model in two stages. In stage one, we train the model for image segmentation only. Concretely, we train on COCO [10] instance segmentation and the target video segmentation dataset without applying any temporal supervision.

In the second stage, we introduce temporal modeling and fine-tune the model from stage one for video segmentation. Unlike CAVIS, DVIS-DAQ, and DVIS++, which freeze the DINOv2-initialized ViT encoder after stage one, we keep fine-tuning the ViT encoder for VidEoMT. We explore fine-tuning the ViT encoder for the CAVIS and DVIS++ baselines in Tabs. 1 and 2 as well, but find that the loss diverges or the memory increases beyond the GPUs' limits. For our VidEoMT, note that fine-tuning the encoder is necessary because our model is encoder-only, meaning that the encoder weights need to be optimized to allow the model to be trained for video segmentation.

A.2. Evaluation

During evaluation, we process videos in a frame-by-frame fashion, as is required for online video segmentation. We evaluate efficiency in terms of FPS and GFLOPs. All metrics are measured on a single NVIDIA H100 GPU using PyTorch 2.7 and CUDA 12.6. We use a batch size of 1 frame to report mean values computed across all frames in the entire validation set. **FPS** is measured using FlashAttention v2 [4] and `torch.compile` [1] with default settings and automatic mixed precision, after 100 warm-up iterations. **FLOPs** are measured with fvcore [13], and reported in GFLOPs (FLOPs $\times 10^{-9}$).

A.3. Visualizations of Model Configurations

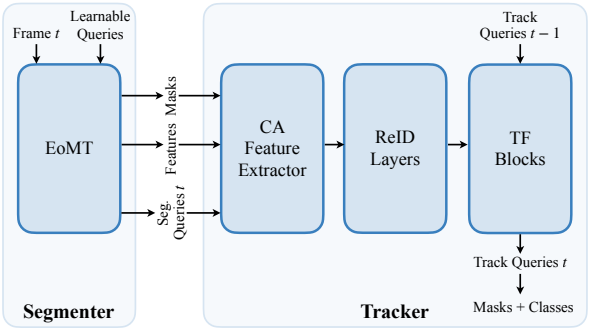
In Sec. 3.3 and Tab. 1, we gradually remove specialized components from the state-of-the-art video segmentation model CAVIS [9], which is visualized in Fig. 2 (left). To provide more details, we additionally illustrate the architectures at intermediate steps (1) to (4) in Fig. A. In the first step, we replace CAVIS's original segmenter – consisting of DINOv2 [16], ViT-Adapter [2], and Mask2Former's pixel decoder and Transformer decoder [3] – with EoMT [8]. In the second step, we remove the context-aware features module and directly forward the segmenter's output queries to the re-identification layers. In the third step, we also remove the re-identification layers, sending the segmenter's output queries directly to the tracker's Transformer blocks. Subsequently, in the fourth step, we discard the tracker altogether, and naively apply EoMT only on a per-frame basis. In this step, temporal association is then obtained in the simplest possible way: we assign all objects predicted from the same query across frames to the same track, without any additional post-hoc temporal matching.

In step (5), which we do not visualize here, we propagate queries by directly feeding the output segmentation queries from frame $t-1$ into the encoder for frame t. Finally, in step (6), we introduce our query fusion design, where propagated queries are fused with learnable queries. The resulting architecture is visualized in Fig. 3 (right).

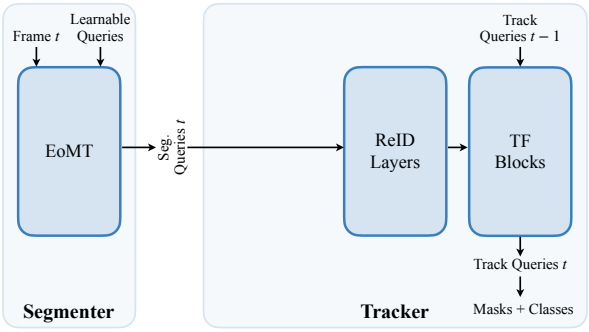
A.4. Hyperparameters

For step (0) in Tab. 1, we report results using the official CAVIS [9] model weights, which we were able to reproduce. For all subsequent steps, we train the models using the same settings as CAVIS with respect to input size, number of iterations, batch size, and number of sampled frames. Specifically, we use a batch size of 8, train on 8 NVIDIA H100 GPUs, and sample 5 frames from a video clip. We train for 160k iterations on YouTube-VIS [19] (all versions) and OVIS [17], for 40k iterations on VIPSeg [15], and for 20k iterations on VSPW [14]. Additionally, all VidEoMT models use $N = 200$ learnable queries with a feature dimension of $D = 1024$.

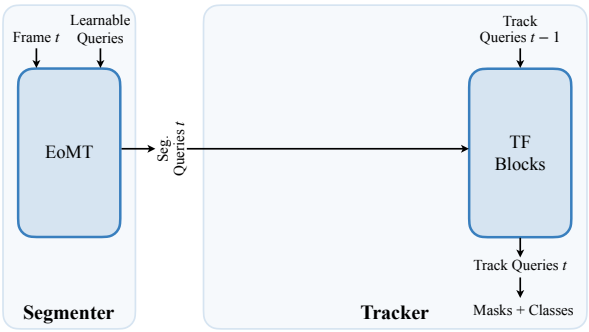
For all experiments that use EoMT as the segmenter, as well as for all experiments with VidEoMT, we keep the optimization strategy identical to that of EoMT. Concretely, we use automatic mixed precision and the AdamW optimizer [11] with a learning rate of 10^{-4}. We apply layer-wise learning rate decay (LLRD) [5] with a factor of 0.6 and polynomial learning rate decay with a power of 0.9. A two-stage linear warm-up strategy is used for all models, including the baselines. Specifically, we first warm up the randomly initialized parameters for 500 iterations while keeping the pre-trained parameters frozen. Then, after 500 iterations, we warm up the pre-trained parameters for 1000 iterations. In both stages, the initial learning rate is set to 0.



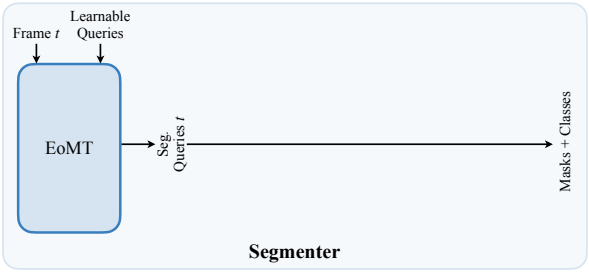
Step (1): w/ EoMT as the Segmenter



Step (2): w/o Context-aware Features



Step (3): w/o Re-identification Layers



Step (4): w/o Tracker Blocks

Figure A. **Removing specialized components.** This figure visualizes the step-by-step removal of complex, specialized components from the CAVIS [9] model, as reported in the results in Tab. 6 of the main manuscript.

To supervise our models, we adopt the same loss functions as Mask2Former [3]. Across all tasks and datasets, we use the cross-entropy (CE) loss for the classification predictions, and the binary cross-entropy (BCE) loss together with Dice loss for segmentation predictions. The total loss is a weighted sum of these components:

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

where λ_{bce}, λ_{dice}, and λ_{ce} are set to 5.0, 5.0, and 2.0, respectively, following Mask2Former [3].

A.5. Architectures of Alternative Approaches

In Tab. 7, we compare VidEoMT with an alternative encoder–decoder architecture that performs temporal modeling in the decoder, with two different temporal modeling approaches: our proposed query fusion and a TrackFormer-based design [12]. As the encoder, we use DINOv2 + ViT-Adapter [2, 16], and as the decoder we use a Transformer that follows the architecture of the Mask2Former Transformer decoder for segmentation. [3]. Concretely, we adopt the standard Mask2Former decoder with 9 layers, each composed of cross-attention, self-attention, and feed-forward blocks, operating with a hidden dimension of 256. To introduce temporal modeling, we feed the track queries and learnable queries into the decoder instead of the encoder's Transformer blocks, which we would do for VidEoMT. At the output of the decoder, the resulting queries are used to predict segmentation masks and classes in the same way as for VidEoMT. For query fusion, we adopt the same approach as described in Sec. 3.4.

The described encoder–decoder approach, which is much less efficient than the encoder-only VidEoMT method (see Tab. 7), somewhat resembles TrackFormer [12], a method for bounding-box multi-object tracking (MOT). TrackFormer also applies temporal modeling by propagating queries into the decoder, but follows a more complex approach to do so. To assess the effectiveness of our query fusion approach compared to TrackFormer's approach, we therefore additionally implement TrackFormer's temporal modeling strategy in the encoder–decoder setting, while staying as close as possible to the original implementation.

Specifically, we make predictions for the first frame using a set of 400 learnable queries. Using these predictions, only the N queries with a classification score $s > 0.8$ are kept and converted into *track queries*. For the next frame, these track queries are concatenated with the 400 original learnable queries, which are then fed to the decoder for that frame. In subsequent frames, the decoder updates the propagated track queries such that they predict the masks for the same objects in the new frames. Again, newly detected queries with scores $s > 0.8$ are added as additional track queries, and non-maximum suppression (NMS) with an IoU threshold of $\sigma_{\text{NMS}} = 0.9$ is applied to remove near-duplicate predictions. Note that this NMS operation is the main rea-

Method	YouTube-VIS 2019 *val* [19]		
	AP	GFLOPs	FPS
No propagation	61.3	565	162
Propagation only	63.9	565	162
Non-object reset	67.8	565	157
TrackFormer	67.7	571	117
Fusion ⇒ VidEoMT	68.6	566	160

Table A. **Query propagation methods.** Comparison of alternative strategies for temporal propagation.

son for the TrackFormer approach's inefficiency compared to VidEoMT's query propagation mechanism. Finally, at each frame, track queries are removed if their score remains below $s < 0.8$ for five consecutive frames, indicating that the object they are tracking has disappeared from the scene.

B. Additional Experiments

B.1. Query Propagation Methods

VidEoMT propagates queries by fusing the learnable queries with the propagated track queries. In Tab. A, we compare this approach with alternative methods to propagate queries.

We start with the *no propagation* approach, the simplest variant, where the model receives only the learnable queries – similar to EoMT – but is fine-tuned for video segmentation. This setting performs the worst, as it lacks any form of explicit temporal modeling.

Next, in the *propagation only* variant, we introduce temporal modeling by directly propagating the output queries from the previous frame into the current frame's encoder. This is step (5) in Tab. 1. However, this approach struggles to detect new objects effectively, as the influence of the learnable queries diminishes over time.

Non-object reset improves over this by replacing a propagated query with a learnable query if it did not predict an object in the previous frame, but this still underperforms the default fusion approach.

Finally, we evaluate the *TrackFormer* approach [12] of only propagating queries for detected objects and introducing new learnable queries to detect new objects. This approach performs slightly worse than our *fusion* approach, but most importantly it is considerably slower because it requires filtering out duplicate detections that should not be propagated. Overall, these results demonstrate that our *fusion* approach is the most accurate and efficient.

B.2. Impact of Model Size

In Tab. 9 of the main manuscript, we report the impact of model size for both VidEoMT and CAVIS. In this section, in Tab. B, we additionally report the results of the EoMT + CAVIS combination, which we also visualize in Fig. 1. Compared to the alternative approach of extending EoMT with a CAVIS tracker, VidEoMT consistently performs better in terms of both efficiency and accuracy across all backbones. This highlights the effectiveness of VidEoMT over the more naive approach of extending EoMT with a state-of-the-art tracker.

Method	Size	AP	Params	GFLOPs	FPS
CAVIS	L	68.9	358M	838	15
EoMT + CAVIS		68.1	328M	699	42
VidEoMT		68.6	318M	566	160
CAVIS	B	59.5	131M	390	18
EoMT + CAVIS		57.4	103M	284	67
VidEoMT		58.2	95M	182	251
CAVIS	S	55.5	57M	251	19
EoMT + CAVIS		50.3	34M	150	93
VidEoMT		52.8	25M	56	294

Table B. **Impact of model size.** VidEoMT performs better as ViT [6] size increases. Evaluated on YouTube-VIS 2019 *val*.

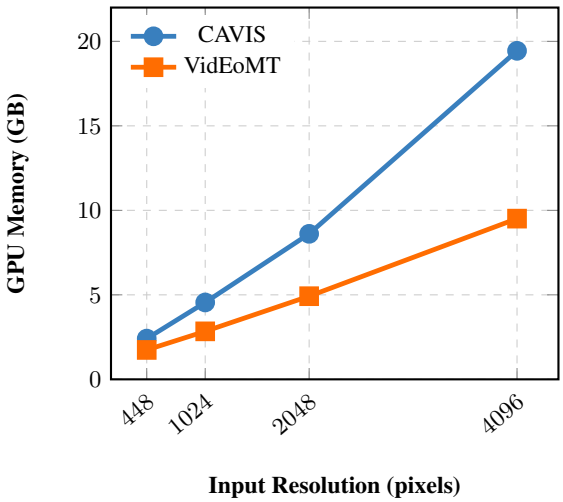


Figure B. **GPU memory consumption of CAVIS and VidEoMT.** Across increasing input resolutions.

B.3. Impact of Pre-training

In Tab. 8 of the main manuscript, we observe that DINOv3 [18] and EVA-02 [7] are slower than DINOv2 [16], despite having a similar number of GFLOPs. Since both DINOv3 and EVA-02 use rotary positional embeddings (RoPE), we attribute a significant part of this slowdown to RoPE, as it introduces additional element-wise operations in the attention layers. When we disable RoPE in these models, we obtain faster models, confirming that RoPE is one of the main sources of the slowdown. Other implementation details may also play a secondary role.

B.4. Comparison of Memory Consumption

Fig. B compares the GPU memory consumption of CAVIS and VidEoMT at different input resolutions. As the resolution increases, the memory usage of both methods grows, but VidEoMT consistently requires substantially less mem-

ory than CAVIS. The gap becomes more pronounced at higher resolutions. For instance, at a resolution of 4096 pixels, CAVIS consumes 19.44 GB of memory, whereas VidEoMT requires only 9.51 GB. This demonstrates that VidEoMT scales more efficiently with increased spatial resolution.

C. Qualitative Results

In Figs. C to E, we visualize the predictions of both CAVIS [9] and VidEoMT for VIS and VPS on the YouTube-VIS 2019 [19], OVIS [17], and VIPSeg [15] datasets.

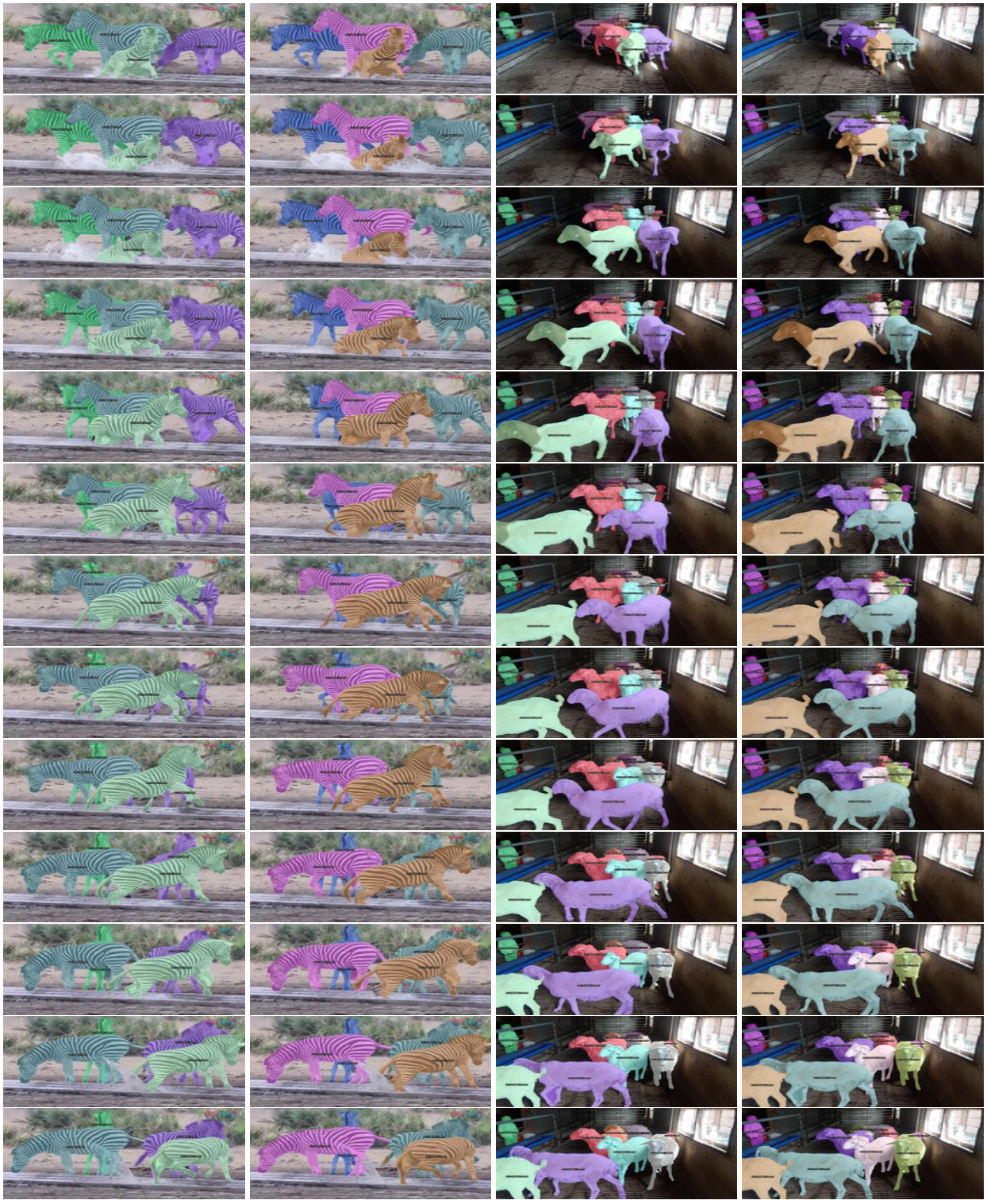
D. Failure Cases

In Figs. F and G, we present failure cases of CAVIS [9] and VidEoMT on the YouTube-VIS 2019 [19] dataset. VidEoMT occasionally fails under long-term occlusions because query propagation is performed only between adjacent frames and because VidEoMT lacks a long-term memory mechanism (row 3 in Fig. F and Fig. G). It also struggles with newly appearing objects when the fixed set of queries has already been assigned to existing instances (row 6 in Fig. G).

However, these limitations also apply to previous state-of-the-art methods, as they typically track objects only across consecutive frames and similarly lack long-term memory mechanisms. For instance, CAVIS also fails under long-term occlusions (row 4 in Fig. G) and when new objects appear (row 3 in Fig. F and rows 4–6 in Fig. G).

CAVIS (15 FPS) **VidEoMT (160 FPS)** CAVIS (15 FPS) **VidEoMT (160 FPS)**

Figure C. **Qualitative results for video instance segmentation.** We compare CAVIS [9] to VidEoMT on the YouTube-VIS 2019 dataset [19].



CAVIS (15 FPS) **VidEoMT (112 FPS)** CAVIS (15 FPS) **VidEoMT (112 FPS)**

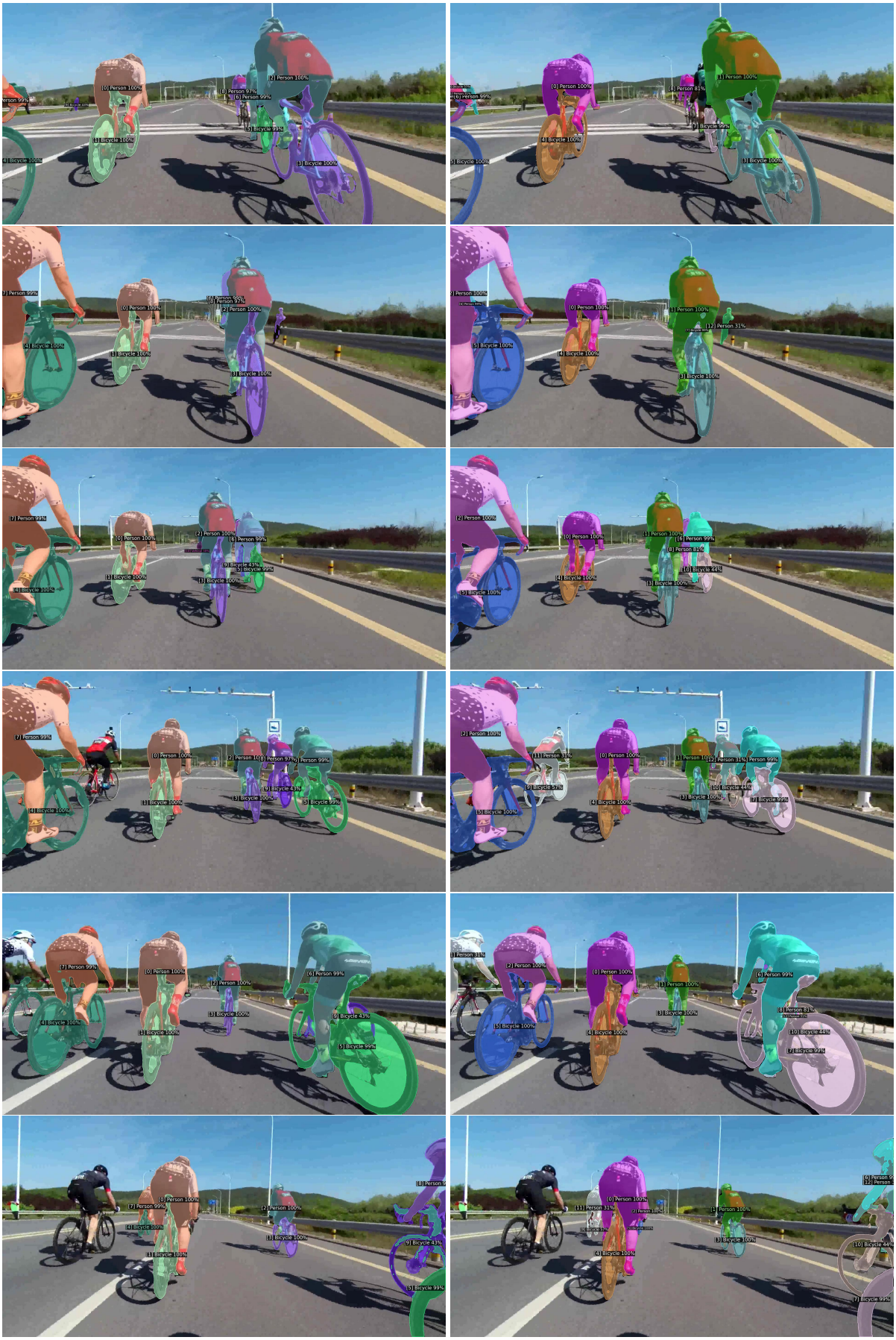
Figure D. **Qualitative results for video instance segmentation.** We compare CAVIS [9] to VidEoMT on the OVIS dataset [17].

CAVIS (10 FPS) **VidEoMT (75 FPS)** CAVIS (10 FPS) **VidEoMT (75 FPS)**

Figure E. **Qualitative results for video panoptic segmentation.** We compare CAVIS [9] to VidEoMT on the VIPSeg dataset [15].

CAVIS (15 FPS) **VidEoMT (112 FPS)**

Figure F. **Failure cases.** We illustrate failure cases of CAVIS [9] and VidEoMT on the OVIS dataset [17]. In row 3, in a highly occluded scene, identity switching occurs between persons for both models. In row 4, both methods fail to detect newly appearing elephants and persons.



CAVIS (15 FPS) **VidEoMT (112 FPS)**

Figure G. **Failure cases.** We illustrate a failure case of CAVIS [9] and VidEoMT on the OVIS dataset [17]. Both methods fail to detect newly appearing cyclists and bicycles in later frames.

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. 1

[2] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision Transformer Adapter for Dense Predictions. In *ICLR*, 2023. 1, 2

[3] 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

[4] Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *ICLR*, 2024. 1

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 2019. 1

[6] 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. 1, 3

[7] 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. 3

[8] Tommie Kerssies, Niccolò Cavagnero, Alexander Hermans, Narges Norouzi, Giuseppe Averta, Bastian Leibe, Gijs Dubbelman, and Daan de Geus. Your ViT is Secretly an Image Segmentation Model. In *CVPR*, 2025. 1

[9] Seunghun Lee, Jiwan Seo, Kiljoon Han, Minwoo Choi, and Sunghoon Im. Context-Aware Video Instance Segmentation. In *ICCV*, 2025. 1, 2, 4, 5, 6, 7, 8, 9

[10] 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. 1

[11] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 1

[12] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. TrackFormer: Multi-Object Tracking with Transformers. In *CVPR*, 2022. 2, 3

[13] Meta Research. fvcore, 2023. 1

[14] Jiaxu Miao, Yunchao Wei, Yu Wu, Chen Liang, Guangrui Li, and Yi Yang. VSPW: A Large-scale Dataset for Video Scene Parsing in the Wild. In *CVPR*, 2021. 1

[15] Jiaxu Miao, Xiaohan Wang, Yu Wu, Wei Li, Xu Zhang, Yunchao Wei, and Yi Yang. Large-Scale Video Panoptic Segmentation in the Wild: A Benchmark. In *CVPR*, 2022. 1, 4, 7

[16] 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. 1, 2, 3

[17] Jiyang Qi, Yan Gao, Yao Hu, Xinggang Wang, Xiaoyu Liu, Xiang Bai, Serge Belongie, Alan Yuille, Philip HS Torr, and Song Bai. Occluded Video Instance Segmentation: A Benchmark. *IJCV*, 130(8):2022–2039, 2022. 1, 4, 6, 8, 9

[18] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025. 3

[19] Linjie Yang, Yuchen Fan, and Ning Xu. Video Instance Segmentation. In *ICCV*, 2019. 1, 3, 4, 5

[20] Tao Zhang, Xingye Tian, Yikang Zhou, Shunping Ji, Xuebo Wang, Xin Tao, Yuan Zhang, Pengfei Wan, Zhongyuan Wang, and Yu Wu. DVIS++: Improved Decoupled Framework for Universal Video Segmentation. *IEEE TPAMI*, 2025. 1

[21] Yikang Zhou, Tao Zhang, Shunping Ji, Shuicheng Yan, and Xiangtai Li. Improving Video Segmentation via Dynamic Anchor Queries. In *ECCV*, 2024. 1