Appendix

A. Implementation Details

Code is provided in the supplementary materials and will be open-sourced.

Training and evaluation datasets.

Video datasets. ***DAVIS17*** [48] is designed for video object segmentation, comprising 150 videos, with 60 allocated for training, 30 for validation, and 60 for testing. Only the first frames of the test set have ground truth foreground masks, so the validation set is used for evaluation. ***YTVOS*** [64] is another dataset for video object segmentation and is significantly larger than DAVIS17. It consists of 4,453 videos that are annotated with 65 object categories. As with DAVIS17, ground truth masks are only available for the first frames of the test and validation sets, and therefore, a fixed 20% of the training set is randomly sampled for the evaluation phase, details are provided in the supplementary material. Additionally, meta information is utilized to ensure objects in the same category have the same class id throughout the dataset for semantic, category-level assessments. Figure 4 shows the distribution of objects in YTVOS.

Image datasets. ***Pascal VOC 2012*** [16] is an object recognition dataset with 20 object categories and one background class. It includes pixel-level segmentation, bounding box, and object class annotations for each image, and has been extensively used as a benchmark for object detection, semantic segmentation, and classification tasks. The dataset is split into three subsets, with 1,464 images allocated for training, 1,449 for validation, and a private testing set. As the dataset has been commonly used as a main reference for recent works in dense self-supervised image segmentation [74, 57, 61], we also use its validation set as one of the evaluation datasets.

Model training. We use batches of size 128 on 1 NVIDIA GeForce RTX 3090, and the optimizer is AdamW [42] with learning rate equal to 1e-4 for the projection head and the backbone's learning rate is 1e-5. We freeze the backbone model except for the last two blocks for fine-tuning. Our model is implemented in torch [47]. We use Faiss [33] for K-Means clustering. We chose to train a ViT-Small [15] image because it has roughly the same number of parameters as a ResNet-50 (21M vs. 23M). The projection head learning rate is 1e-4 and the backbone's learning rate is 1e-5. The projection head consists of three linear layers with hidden dimensionality of 2048 and Gaussian error linear units as activation function [26]. We set the temperature to 0.1 and use Adam as an optimizer with a cosine weight decay schedule. The augmentations used are random color-jitter, Gaussian blur, grayscale, and random cropping.

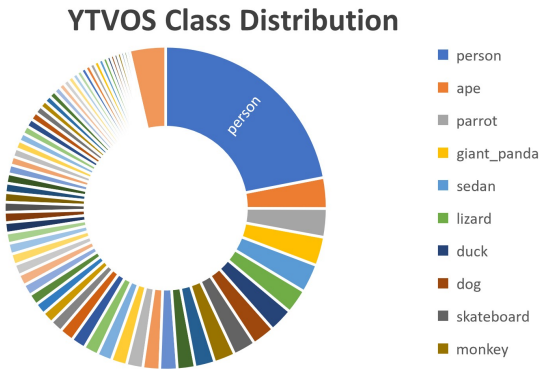

Figure 4: The distribution of classes in YTVOS. Some of the more dominant classes are labeled.

```
# mm : torch.mm
# exp : torch.exp
# bmm : torch.bmm
# Reshape : In-place operation to change the
    input shape
# Normalize : torch.Normalize
# F[i] : Shows the ith feature map
# C-Map[i] : Shows the ith cluster map

prev_feat = []  # (nmb-context, dim, h*w)
prev_maps = []
For i in range(N-1):
    prev_feat.append(F[i])
    prev_maps.append(C-Map[i])
src_feat = Stack(prev_feat)
trgt_feat = F[N] # (1, dim, h*w)
trgt_feat= Normalize(trgt_feat, dim=1, p=2)
src_feat = Normalize(src_feat, dim=1, p=2)
aff = exp(bmm(trgt_feat, src_feat) / 0.1)
Reshape(aff, (nmb-context * h*w, h*w))
aff = aff / sum(aff, keepdim=True, axis=0)
aff = mask-neighborhood(aff)
prev_maps = Stack(prev_maps)  # (nmb-context, C,
    h, w)
Reshape(prev_maps, (C, nmb-context*h*w))
trgt_cmap = mm(prev_maps, aff)
```

Listing 1: **FF component**. The pytorch implementation of **FF** is shown.

Evaluation details. Since we evaluate the pre-GAP *layer4* features or the spatial tokens, their output resolution does not match the mask resolution. To fix that, we bilinearly interpolate before applying the linear head; or directly interpolate the clustering results by nearest neighbor upsampling. For a fair comparison between ResNets and ViTs,

Algorithm 1 Evaluation Pipeline Pseudocode. M is the model, C is the clustering algorithm, MA is the matching algorithm by which the clusters are scored, and GT is the given ground-truth.

```
input = input.reshape(bs * n_f, c, h, w)
F_b = M(input)
F_b = F.reshape(bs, n_f, num-patch, dim)
score_list = []
if Per frame then
    for F_c In F_b do
        for F_f In F_c do
            C_Map = C(F_f)
            score = MA(C_Map, GT_f)
            score_list.append(score)
        end for
    end for
else if Per clip then
    for F_c In F_all do
        C_Maps = C(F_c)
        score = MA(C_Maps, GT_c)
        score_list.append(score)
    end for
else if Per dataset then
    C_Maps = C(F_b)
    score = MA(C_Maps, GT_b)
    score_list.append(score)
end if
return(score_list.mean())
```

we use dilated convolution in the last bottleneck layer of the ResNet such that the spatial resolution of both network architectures match (28x28 for 448x448 input images). All overclustering results were computed using downsampled 100x100 masks to speed up the Hungarian matching as we found that the results do not differ from using full-resolution masks.

B. Evaluation Protocol for Unsupervised Video Object Semantic Segmentation

Here, we provide details for the evaluation protocols for unsupervised video multi-label object segmentation. To be consistent with the image domain [74], a clustering algorithm is applied to the features extracted from frozen encoders to craft dense assignment maps of pseudo-labels. To produce scores, based on each evaluation protocol, the crafted maps are matched with the ground truth, and their MIOU is reported. Suppose the matching algorithm is specified by $M(\text{labels}, \text{ground-truth})$, clustering algorithm by K, dataset features by $F \in R^{N \times n\_f \times d \times h \times w}$, where N, n_f, d, h, and w stand for dataset size, number of frames per clip, feature dimension, and feature spatial resolutions, respectively. we introduce three evaluation protocols that are specific to the video domain.

Per frame evaluation (*F*).

$$F_{\text{frame}}[i,j] = F[i,j] \tag{10}$$

$$\text{score} = \frac{1}{N \times n\_f} \sum_{i=1}^{N} \sum_{j=1}^{n\_f} \text{MIOU}(M(K(F_{\text{frame}}[i,j]), \text{GT}[i,j])) \tag{11}$$

this measures a basic alignment of a given feature map with the ground-truth.

Per clip evaluation (*C*).

$$F_{\text{clip}}[i] = (F[i,1], \cdots, F[i,nf]) \tag{12}$$

$$\text{score} = \frac{1}{N} \sum_{i=1}^{N} \text{MIOU}(M(K(F_{\text{clip}}[i]), \text{GT}[i])) \tag{13}$$

This evaluation tests whether the assigned pseudo-labels remain consistent over time for each clip.

Per dataset evaluation (*D*).

$$F_{\text{dataset}} = (F[1,1], \cdots, F[N,n\_f]) \tag{14}$$

$$\text{score} = \text{MIOU}(M(K(F_{\text{dataset}}), \text{GT})) \tag{15}$$

This evaluation measures the most difficult ability of generating not only temporally stable features of objects across time but across videos.

C. Additional Experiments

To provide a complete evaluation of our method compared to the baseline on Pascal VOC [16], we show the per-class performance in Figure 6. As the figure shows, we improve the class "person" by more than 40%, which could be beacuse of the high number of such objects in YTVOS as Figure 4 shows. The classes "cat" and "dog" also show a significant improvement since they are of the further dominant classes after "person".

Unsupervised video object segmentation and tracking. Although such methods are designed for salient object detection or unsupervised mask propagation, and not specifically for unsupervised semantic segmentation, we evaluate their performance on our proposed evaluation protocols to highlight their strengths and limitations. The results are shown in Table 9. As it is shown, TIMET outperforms such methods on all the evaluation protocols by a margin between 12% to 24%. Not being specifically designed for semantic segmentation tasks may explain their inferior performance.

Method	*F*	*C*	*D*
DUL [1]	28.2	27.4	2.4
Motion Grp [66]	32.0	30.7	1.5
TIMET (ours)	**56.5**	**55.5**	**14.1**

Table 9: **Comparison to video unsupervised object segmentation methods.** Evaluation on DAVIS with K=GT.

Comparing to unsupervised video object segmentation and tracking. Although such methods are designed for salient object detection or unsupervised mask propagation, and not specifically for unsupervised semantic segmentation, we evaluate their performance on our proposed benchmark to highlight their strengths and limitations. We conducted a comprehensive comparison of our method with state-of-the-art techniques, such as Motion-Grp [66] and DUL [1], which aim to learn unsupervised features for propagating a given first frame's mask in test time or separating foreground from background using motion flows, as previously mentioned. To ensure consistency, we trained and tested all models on DAVIS, a widely used benchmark. Results in Table 9 demonstrate that our method outperforms these techniques across all reported metrics. It is worth noting that these methods were not specifically designed for semantic segmentation tasks, which may explain their inferior performance.

Analyzing per class results. In Figure 6 and 5 the per class improvements on Pascal VOC are reported. As the figures show, for the classes that make over 95% of the number of objects existing in YTVOS, the performance almost always improves considerably. The reason that the class "bird" does not behave the same as the others might be due to the small size of this object in YTVOS.

Component Contributions. In Table 10, we show the effect of using different components on the Pascal clustering results. As it is shown, TIMETimproves DINO [9] by 12%. The results are improved by another 18% after applying CBFE [74] to the features, showing high overclustering performance on this dataset. The number of clusters used for CBFE is 300 for this experiment.

Comparing different propagators. Complementary to the previous ablations, in Table 11, we conduct a detailed study on the performance of forwarding foreground masks using various temporal intervals and comparing the use of optical flow [53] with our Feature-Forwarder (FF). We find that across all values of δt, FF outperforms flow by a large margin of +10% or more. This superiority has the added benefit of our FF module not having to expensively compute flow but instead reusing the activations that are processed for the clustering step.

	MIOU
K=150	48.2
DINO	4.6
+TimeT	16.5
+CBFE	34.5

Table 10: **Component contributions.** We show the gains that each individual component brings for Pascal VOC segmentation and K=21.

δt	GT_0	Flow	FF
0.1s	26.4	29.8	39.8
0.2s	25.7	28.5	**40.3**
0.4s	24.5	26.1	40.1
0.8s	21.8	22.7	37.5

Table 11: Comparing FF with other forwarding methods. The numbers of reported on DAVIS validation set.

D. Additional Visualisations

Clustering with K=GT. In Figure 8, we show some qualitative results on Pascal VOC when K is set to the number of ground-truth objects, similar to Figure 3 in the paper. While the method is trained on YTVOS with a temporal loss, it obtains strong performances on an image segmentation dataset yielding high class consistencies (indicated by the segmentation colors) and tight borders.

We also have provided further qualitative results with the same setting on DAVIS and YTVOS in Figure 7 and the attached HTML file. As the videos show, we get considerably more consistent and structured visualizations compared to DINO [9] and Leopart [74]. This further supports the effectiveness of temporal fine-tuning compared to the models solely trained on images.

Overclustering results by merging 500 clusters using ground-truth labels. In the attached HTML file, we show the visualizations of our method on Pascal in the overslutering setting as well. As depicted, objects from different classes can be segmented precisely with different colors, showing that the learned patch features are semantic.

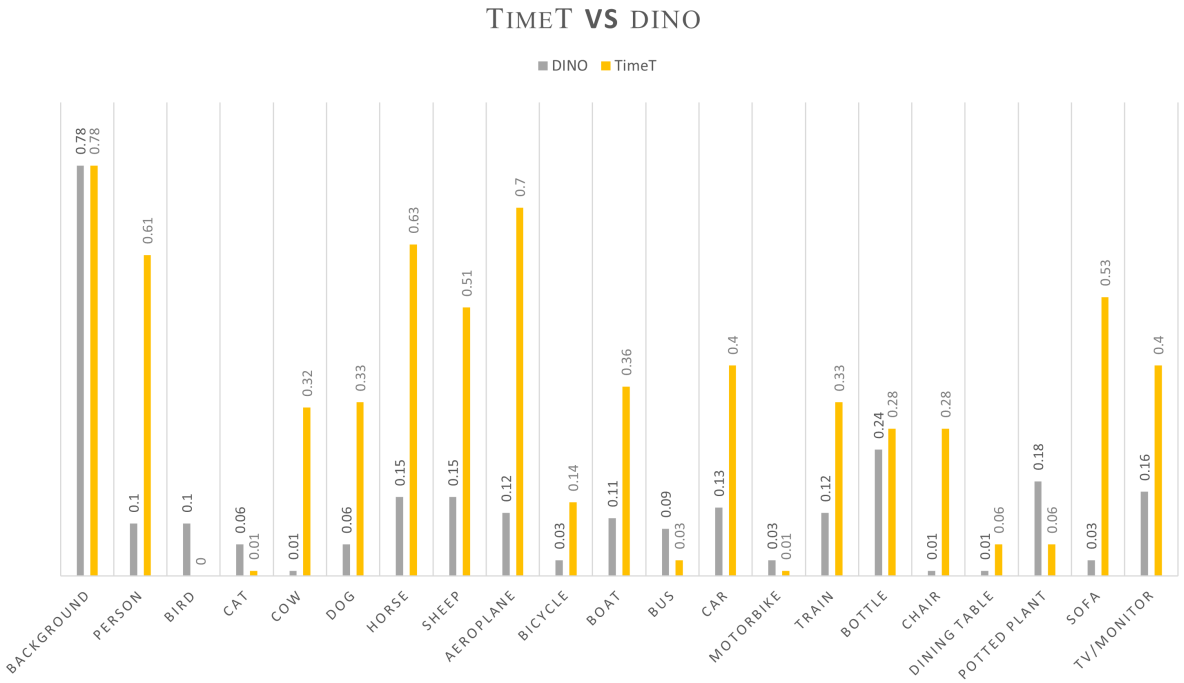

Figure 5: The per class performance of DINO and TIMET is shown for the clustering experiment with K=GT. Cluster-based foreground extraction [74] has been applied to both methods. As it is seen, this paper almost always improves the baseline performance for this evaluation as well. Pascal VOC is used in this experiment.

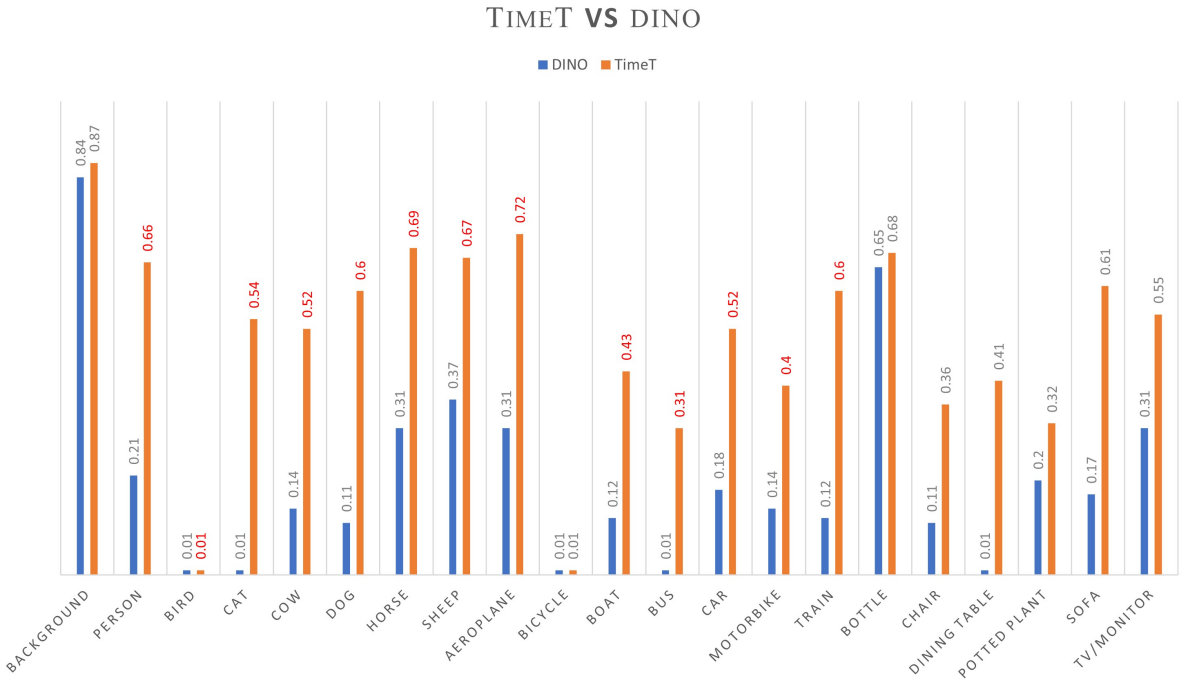

Figure 6: The per class performance of DINO and TIMET is shown for the overclustering experiment with K=500. As it is seen, this paper consistently improves the baseline performance. The numbers for the dominant shared classes between YTVOS and Pascal VOC are shown by the color red.

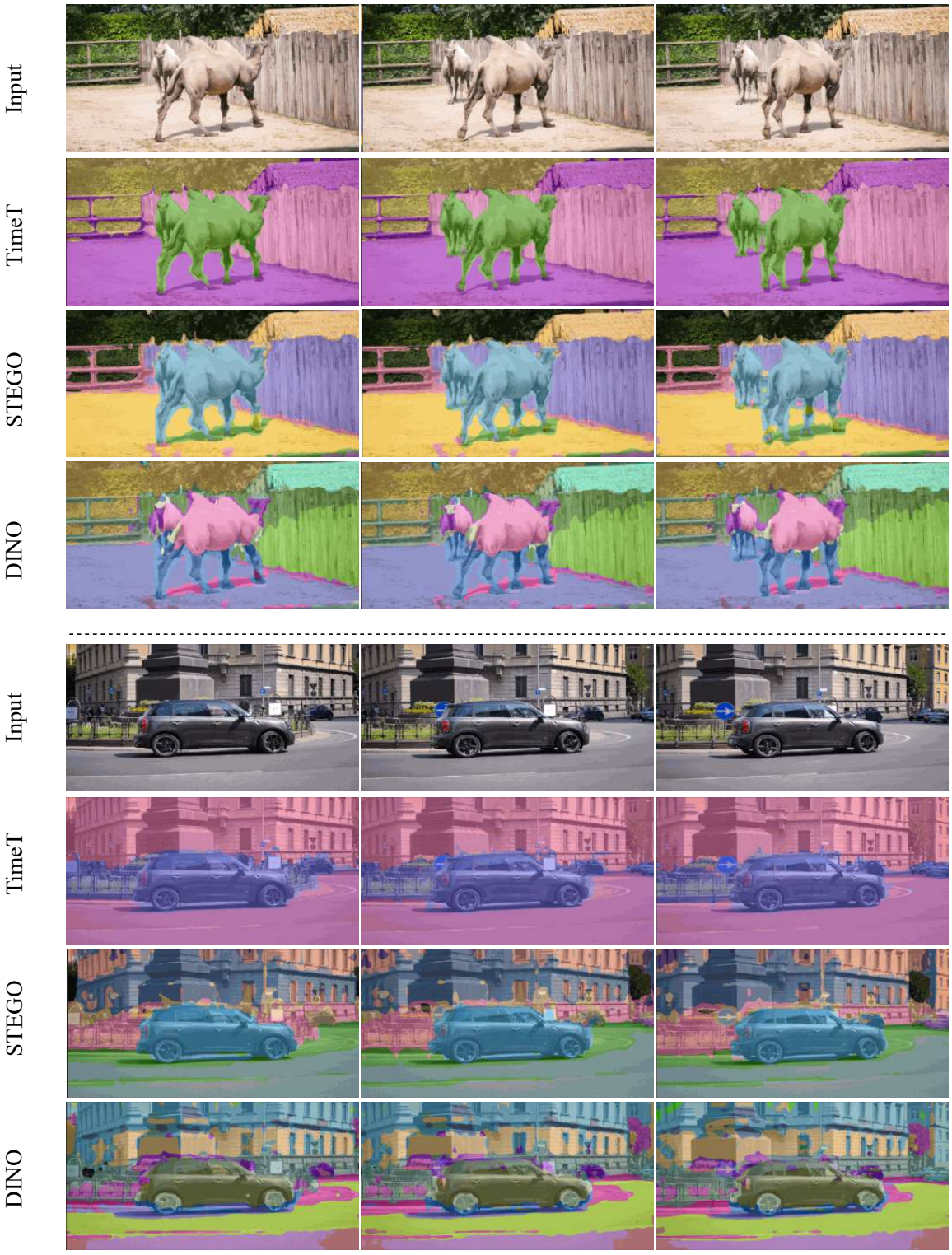

Figure 7: **TIMET segmentations on DAVIS with K=GT.** Here, we compare the performance of DINO, STEGO, and TIMET on the task of unsupervised video semantic segmentations. TIMET has a clear advantage over both DINO and STEGO in terms of providing tight segmentation boundaries and specifying different objects with different category IDs. Different colors in the figure specify different IDs.

Figure 8: **TimeT segmentations on Pascal VOC with K=21.** We use CBFE [74] to focus on the foreground objects. While the method is trained on YTVOS with a temporal loss, it obtains strong performances on an image segmentation dataset yielding high class consistencies (indicated by the segmentation colors) and tight borders.

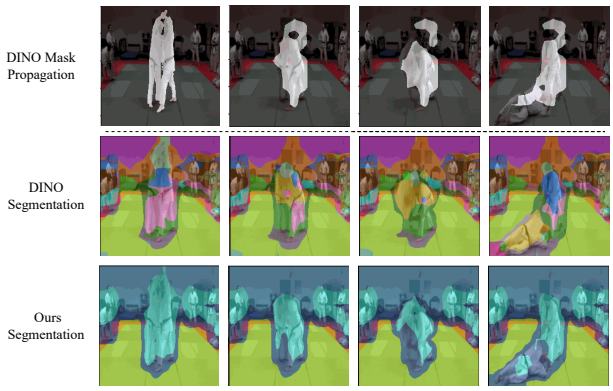

Figure 9: DINO's features jump around across time, leading to inconsistent cluster maps. Our proposed TIMET-trained model observes more temporal consistency.