Reducing the Annotation Effort for Video Object Segmentation Datasets: Supplemental Material

1. Video Instance Segmentation Experiments

Video Instance Segmentation (VIS) [5] is a recently introduced computer vision task related to VOS. Unlike VOS, for VIS a pre-defined set of categories is used and no first-frame ground truth mask is given. All objects of the pre-defined categories need to be detected, segmented, classified into the right category, and tracked over time. The performance is measured by the Average Precision ($\mathcal{AP}$), and Average Recall ($\mathcal{AR}$) measures which are also used for image instance segmentation. However, for VIS, these measures are calculated on the whole track level (for details see [5]).

For VIS, so far the only dataset is YouTube-VIS [5] which features 40 categories, 131k instance masks, and 2,883 videos. In order to show that our method for creating pseudo-labels is not only effective for the VOS task, we also applied it to VIS, and used these labels for training STEm-Seg [1], a state-of-the-art VIS method. STEm-Seg uses a single-stage network to learn a spatio-temporal pixel embedding to cluster instances over an entire video clip. We used the code provided by the authors to train STEm-Seg on our Box2Seg labels created for YouTube-VIS. The results are shown in Table 1. It can be seen that when using the labels created by Box2Seg based on only the bounding boxes of YouTube-VIS, the $\mathcal{AP}$ decreases from 34.6 to 30.5, *i.e.* by 4.1 percentage points. When adding only a single manually annotated mask per object, and using it for fine-tuning Box2Seg as done in the main paper for YouTube-VOS, the gap decreases significantly to only 1.3 percentage points. This demonstrates that our pseudo-label generation method is also applicable for different datasets, tasks, and methods trained on them.

2. Effect of Inference Frame Rate

The TAO-VOS validation set is only annotated at 1 frame per second, while the video data is available at a higher frame rate of around 30 frames per second. Similar to the evaluation protocol for YouTube-VOS, an algorithm can either run only on the annotated frames, or it can use all frames. Intuitively, tracking objects at a higher frame-rate should be easier, because there is less motion between two adjacent frames. For each method we tried both variants (see Table 2). Interestingly, only LWL can benefit from the higher frame rate, while STM-VOS and CFBI produced better results when run only on the annotated frames. The difference is especially strong for CFBI, where using all frames is 7.6 percentage points worse than when just evaluating on the annotated frames. By qualitative inspection, we found that for STM-VOS and CFBI at a high frame rate the predicted mask can get slightly smaller from frame to frame or it can grow. In both cases, small errors accumulate over time and lead to worse results. When using the low frame rate, it is harder to transfer information between the frames, because there is more motion, but at the same time, small errors cannot easily accumulate over time. We found that LWL produces masks which are more stable over time, and hence it can benefit from the higher frame rate.

3. Implementation Details of Fine-Tuning

Here we provide implementation details of fine-tuning the three considered methods on the TAO-VOS training set. Note that we mainly mention the differences to the default settings of the provided code for the considered methods.

For fine-tuning STM-VOS we used the robust loss function, *i.e.* the partially Huberised cross entropy loss described in the main paper. Fine-tuning was then done using the Adam optimizer [3] with a learning rate of 10^{-7} for 50 epochs (the main training on YouTube-VOS was done with Adam with a learning rate of 10^{-6} for 300 epochs).

CFBI and LWL use different loss functions which are not as straightforward to adapt for robustness, hence we keep their original loss functions.

For CFBI, we fine-tuned on the TAO-VOS training set for 5,000 steps using a learning rate of 10^{-3} (the main training on YouTube-VOS used a learning rate of 10^{-2}).

For LWL, in the default setup, the learning rate differs for different weights and training is done for 70 epochs. For fine-tuning on the TAO-VOS training set, we reduced the learning rate by a factor of 10 and fine-tuned for 20 epochs.

Method	YouTube-VIS validation set				
	$\mathcal{AP}$	$\mathcal{AP}$@50	$\mathcal{AP}$@75	$\mathcal{AR}$@1	$\mathcal{AR}$@10
STEm-Seg [1]	34.6	55.8	37.9	34.4	41.6
STEm-Seg (Box2Seg labels)	30.5 (-4.1)	50.9	33.5	29.8	36.4
STEm-Seg (Box2Seg fine-tuned labels)	33.3 (-1.3)	53.1	36.0	33.1	39.5

Table 1. Results for Video Instance Segmentation (VIS) on the YouTube-VIS validation set. Here we use the state-of-the-art method STEm-Seg and train it with different sets of labels. $\mathcal{AP}$@50 and $\mathcal{AP}$@75 denote average precision at an IoU threshold of 50% or 75%, respectively, and $\mathcal{AP}$ is averaged over different thresholds. $\mathcal{AR}$ denotes the maximum recall achieved by a fixed number of segmented instances per video [5].

Method	TAO-VOS val		
	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
STM [4], annotated frames	63.8	61.5	66.1
STM [4], all frames	60.6 (-3.2)	58.9	62.3
CFBI [6], annotated frames	66.3	63.8	68.8
CFBI [6], all frames	58.7 (-7.6)	56.6	60.8
LWL [2], annotated frames	55.6	53.5	57.8
LWL [2], all frames	59.5 (+3.9)	56.7	62.4

Table 2. Effect of performing inference only on the annotated frames (1 frame per second), or evaluating on all frames. Only LWL benefits from the higher frame rate, and STM and CFBI perform better with the lower frame rate.

References

[1] Ali Athar, Sabarinath Mahadevan, Aljoša Ošep, Laura Leal-Taixé, and Bastian Leibe. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *ECCV*, 2020.

[2] Bhat Goutam, Felix Järemo Lawin, Martin Danelljan, Andreas Robinson, Michael Felsberg, Luc Van Gool, and Radu Timofte. Learning what to learn for video object segmentation. In *ECCV*, 2020.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[4] S. Wug Oh, J.-Y. Lee, N. Xu, and S. Joo Kim. Video object segmentation using space-time memory networks. In *ICCV*, 2019.

[5] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *ICCV*, 2019.

[6] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by foreground-background integration. In *ECCV*, 2020.