Online Adaptation of Convolutional Neural Networks for Video Object Segmentation

Paul Voigtlaender
voigtlaender@vision.rwth-aachen.de
Bastian Leibe
leibe@vision.rwth-aachen.de

Computer Vision Group
Visual Computing Institute
RWTH Aachen University
Germany

Abstract

We tackle the task of semi-supervised video object segmentation, *i.e*. segmenting the pixels belonging to an object in a video using the ground truth pixel mask for the first frame. We build on the recently introduced one-shot video object segmentation (*OSVOS*) approach which uses a pretrained network and fine-tunes it on the first frame. While achieving impressive performance, at test time *OSVOS* uses the fine-tuned network in unchanged form and is not able to adapt to large changes in object appearance. To overcome this limitation, we propose Online Adaptive Video Object Segmentation (*OnAVOS*) which updates the network online using training examples selected based on the confidence of the network and the spatial configuration. Additionally, we add a pretraining step based on objectness, which is learned on PASCAL. Our experiments show that both extensions are highly effective and improve the state of the art on DAVIS to an intersection-over-union score of 85.7%.

1 Introduction

Visual object tracking is a fundamental problem in computer vision with many applications including video editing, autonomous cars, and robotics. Recently, there has been a trend to move from bounding box level to pixel level tracking, mainly driven by the availability of new datasets, in particular DAVIS [34]. In our work, we focus on semi-supervised video object segmentation (VOS), *i.e*. the task of segmenting the pixels belonging to a generic object in the video using the ground truth pixel mask of the first frame.

Recently, deep learning based approaches, which often utilize large classification datasets for pretraining, have shown extremely good performance for VOS [7, 20, 24, 35] and the related tasks of single-object tracking [5, 18, 31] and background modeling [2, 6, 44]. In particular, the one-shot video object segmentation (*OSVOS*) approach introduced by Caelles *et al*. [7], has shown very promising results for VOS. This approach fine-tunes a pretrained convolutional neural network on the first frame of the target video. However, since at test time *OSVOS* only learns from the first frame of the sequence, it is not able to adapt to large changes in appearance, which might for example be caused by drastic changes in viewpoint.

While online adaptation has been used with success for bounding box level tracking (*e.g*. [14, 23, 27, 31, 43]), its use for VOS [3, 4, 10, 32] has received less attention, especially in the context of deep learning. We thus propose Online Adaptive Video Object

Published as a conference paper at BMVC 2017.

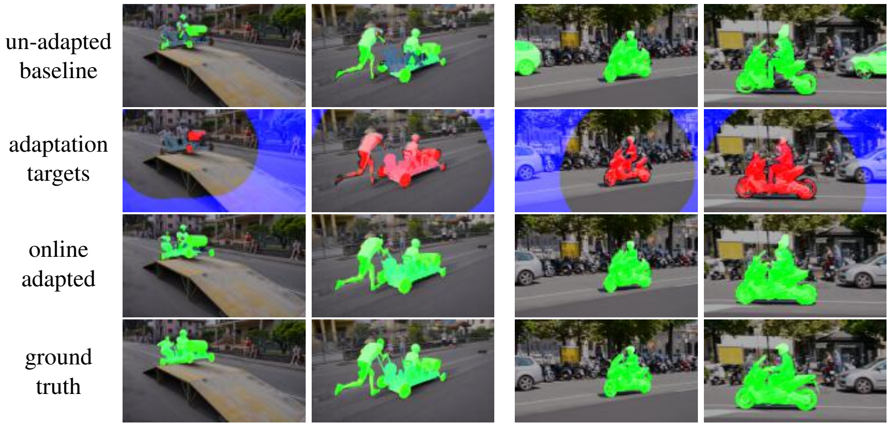


Figure 1: Qualitative results on two sequences of the DAVIS validation set. The second row shows the pixels selected as positive (red) and negative (blue) training examples. It can be seen that after online adaptation, the network can deal better with changes in viewpoint (left) and new objects appearing in the scene (the car in the right sequence).

Segmentation (*OnAVOS*), which updates a convolutional neural network based on online-selected training examples. In order to avoid drift, we carefully select training examples by choosing pixels for which the network is very certain that they belong to the object of interest as positive examples, and pixels which are far away from the last assumed pixel mask as negative examples (see Fig. 1, second row). We further show that naively performing online updates on every frame quickly leads to drift, which manifests in strongly degraded performance. As a countermeasure, we propose to mix in the first frame (for which the ground truth pixel mask is known) as additional training example during online updates.

Our contributions are the following: We introduce *OnAVOS*, which uses online updates to adapt to changes in appearance. Furthermore, we adopt a more recent network architecture and an additional objectness pretraining step [20, 21] and demonstrate their effectiveness for the semi-supervised setup. We further show that *OnAVOS* significantly improves the state of the art on two datasets.

2 Related Work

Video Object Segmentation. A common approach of many classical video object segmentation (VOS) methods is to reduce the granularity of the input space, *e.g.* by using superpixels [8, 15], patches [12, 38], or object proposals [33]. While these methods significantly reduce the complexity of subsequent optimization steps, they can introduce unrecoverable errors early in the pipeline. The obtained intermediate representations (or directly the pixels [30]) are then used for either a global optimization over the whole video [30, 33], over parts of it [15], or using only the current and the preceding frame [8, 12, 38].

Recently, neural network based approaches [7, 20, 24, 35] including *OSVOS* [7] have become the state of the art for VOS. Since *OnAVOS* is built on top of *OSVOS*, we include a detailed description in Section 3. While *OSVOS* handles every video frame in isolation, we expect that incorporating temporal context should be helpful. As a step in this direction, Perazzi *et al.* [35] propose the *MaskTrack* method, in which the estimated segmentation mask from the last frame is used as an additional input channel to the neural network, enabling it to use temporal context. Jampani *et al.* [22] propose a video propagation network (*VPN*) which applies learned bilateral filtering operations to propagate information across video frames. Furthermore, optical flow has been used as an additional temporal cue in conjunction with deep learning in the semi-supervised [24, 35] and unsupervised setting [40], in which the ground truth for the first frame is not available. In our work, we focus on including context information implicitly by adapting the network online, *i.e.* we store temporal context information in the adapted weights of the network.

Recently, Jain *et al.* [21] proposed to train a convolutional neural network for pixel objectness, *i.e.* for deciding for each pixel whether it belongs to an object-like region. In another paper, Jain *et al.* [20] showed that using pixel objectness is helpful in the unsupervised VOS setting. We adopt pixel objectness as a pretraining step for the semi-supervised setting based on the one-shot approach.

The current best result on DAVIS is obtained by *LucidTracker* from Khoreva *et al.* [24], which extends *MaskTrack* by an elaborate data augmentation method, which creates a large number of training examples from the first annotated frames and reduces the dependence on large datasets for pretraining. Our experiments show that our approach achieves better performance using only conventional data augmentation methods.

Online Adaptation. For bounding box level tracking, Kalal *et al.* [23] introduced the *Tracking-Learning-Detection (TLD)* framework, which tries to detect errors of the used object detector and to update the detector online to avoid these errors in the future. Grabner and Bischof [14] used an online version of AdaBoost [13] for multiple computer vision tasks including tracking. Nam and Han [31] proposed a *Multi-Domain Network* (*MDNet*) for bounding box level tracking. *MDNet* trains a separate domain-specific output layer for each training sequence and at test time initializes a new output layer, which is updated online together with two fully-connected layers. To this end, training examples are randomly sampled close to the current assumed object position, and are used as either positive or negative targets, based on their classification scores. This scheme of sampling training examples online has some similarities to our approach. However, our method works on the pixel level instead of the bounding box level and, in order to avoid drift, we take special care to only select training examples online for which we are very certain that they are positive or negative examples. For VOS, online adaptation is less well explored; mainly classical methods like online-updated color and/or shape models [3, 4, 32] and online random forests [10] have been proposed.

Fully Convolutional Networks for Semantic Segmentation. Fully Convolutional Networks (FCNs) for semantic segmentation have been introduced by Long *et al.* [29]. The main idea is to repurpose a network initially designed for classification for semantic segmentation by replacing the fully-connected layers with 1×1 convolutions, and by introducing skip connections which help capture higher resolution details. Variants of this approach have since been widely adopted for semantic segmentation with great success (*e.g.* ResNets by He *et al.* [17]).

Recently, Wu *et al.* [45] introduced a ResNet variant with fewer but wider layers than the original ResNet architectures [17] and a simple approach for segmentation, which avoids some of the subsampling steps by replacing them by dilated convolutions [47] and which does not use any skip connections. Despite the simplicity of their architecture for segmentation, they obtained outstanding results across multiple classification and semantic segmentation datasets, which motivates us to adopt their architecture.

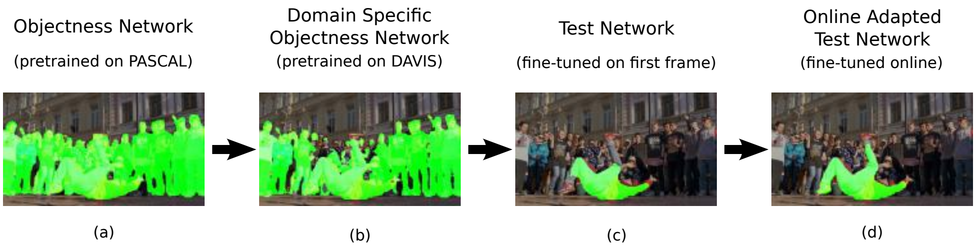


Figure 2: The pipeline of *OnAVOS*. Starting from pretrained weights, the network is first pretrained for objectness on PASCAL (a). Afterwards we pretrain on DAVIS to incorporate domain specific information (b). During test time, we fine-tune on the first frame, to obtain the test network (c). On the following frames, the network is then fine-tuned online to adapt to the changes in appearance (d).

3 One-Shot Video Object Segmentation

OnAVOS (see Fig. 2 for an overview) builds upon the recently introduced one-shot video object segmentation (*OSVOS*) approach [7], but introduces pretraining for pixel objectness [21] as a new component, adopts a more recent network architecture, and incorporates a novel online adaptation scheme, which is described in detail in Section 4.

Base Network. The first step of *OnAVOS* is to pretrain a base network on large datasets (*e.g.* ImageNet [9] for image classification) in order to learn a powerful representation of objects, which can later be used as a starting point for the video object segmentation (VOS) task.

Objectness Network. In a second step, the network is further pretrained for pixel objectness [21] using a binary cross-entropy loss. In order to obtain targets for foreground and background, we use the PASCAL [11] dataset and map all 20 annotated classes to foreground and all other image regions are treated as background. As demonstrated by Jain *et al.* [20], the resulting objectness network alone already performs well on DAVIS, but here we use objectness only as a pretraining step.

Domain Specific Objectness Network. The objectness network was trained on the PASCAL dataset. However, the target dataset on which the VOS should be performed may exhibit different characteristics, *e.g.* a higher resolution and less noise in the case of DAVIS. Hence, we fine-tune the objectness network using the DAVIS training data and obtain a domain specific objectness network. The DAVIS annotations do not directly correspond to objectness, as usually only one object out of possibly multiple is annotated. However, we argue that the learned task here is still similar to general objectness, since in most sequences of DAVIS the number of visible objects is relatively low and the object of interest is usually relatively large and salient. Note that *OSVOS* trained the base network directly on DAVIS without objectness pretraining on PASCAL. Our experiments show that both steps are complementary.

Test Network. After the preceding pretraining steps, the network has learned a domain specific notion of objectness, but during test time, it does not know yet which of the possibly multiple objects of the target sequence it should segment. Hence, we fine-tune the pretrained network on the ground truth mask of the first frame, which provides it with the

Algorithm 1 Online Adaptive Video Object Segmentation (*OnAVOS*)

```
Input: Objectness network $\mathcal{N}$, positive threshold $\alpha$, distance threshold $d$, total online steps $n_{online}$, current frame steps $n_{curr}$
1:  Fine-tune $\mathcal{N}$ for 50 steps on $frame(1)$
2:  $lastmask \leftarrow ground\_truth(1)$
3:  for $t = 2 \dots T$ do
4:      $lastmask \leftarrow erosion(lastmask)$
5:      $dtransform \leftarrow distance\_transform(lastmask)$
6:      $negatives \leftarrow dtransform > d$
7:      $posteriors \leftarrow forward(\mathcal{N}, frame(t))$
8:      $positives \leftarrow (posteriors > \alpha) \setminus negatives$
9:      if $lastmask \neq \emptyset$ then
10:         interleaved:
11:             Fine-tune $\mathcal{N}$ for $n_{curr}$ steps on $frame(t)$ using $positives$ and $negatives$
12:             Fine-tune $\mathcal{N}$ for $n_{online} - n_{curr}$ steps on $frame(1)$ using $ground\_truth(1)$
13:     end if
14:     $posteriors \leftarrow forward(\mathcal{N}, frame(t))$
15:     $lastmask \leftarrow (posteriors > 0.5) \setminus negatives$
16:     Output $lastmask$ for frame $t$
17: end for
```

identity and specific appearance of the object of interest and allows it to learn to ignore the background. This one-shot step has been shown to be very effective for VOS [7], which we also confirm in our experiments. However, the first frame does not provide enough information for the network to adapt to drastic changes in appearance or viewpoint. In these cases, our online adaptation approach (see Section 4) is needed.

Network Architecture. While *OSVOS* used a variant of the well-known VGG network [39], we choose to adopt a more recent network architecture which incorporates residual connections. In particular, we adopt model A from Wu *et al.* [45], which is a very wide ResNet [17] variant with 38 hidden layers and roughly 124 million parameters. The approach for segmentation is very simple, as no upsampling mechanism or skip connections are used. Instead, downsampling by a factor of two using strided convolutions is performed only three times. This leads to a loss of resolution by a factor of eight in each dimension, following which the receptive field is increased using dilated convolutions [47] at no additional loss of resolution. Despite its simplicity, this architecture has shown excellent results both for classification (ImageNet) and segmentation (PASCAL) tasks [45]. When applying it for segmentation, we bilinearly upsample the pixelwise posterior probabilities to the initial resolution before thresholding with 0.5.

We use the weights provided by Wu *et al.* [45], which were obtained by pretraining on ImageNet [9], Microsoft COCO [28], and PASCAL [11], as a very strong initialization for the base network. We then replace the output layer with a two-class softmax. As loss function, we use the bootstrapped cross-entropy loss function [46], which takes the average over the cross-entropy loss values only over a fraction of the hardest pixels, *i.e.* pixels which are predicted worst by the network, instead of all pixels. This loss function has been shown to work well for unbalanced class distributions, which also commonly occur for VOS due to the dominant background class. In all our experiments, we use a fraction of 25% of the hardest pixels and optimize this loss using the Adam optimizer [25]. In our evaluations, we separate the effect of the network architecture from the effect of the algorithmic improvements.

4 Online Adaptation

Since the appearance of the object of interest changes over time and new background objects can appear, we introduce an online adaptation scheme to adapt to these changes (see Algorithm 1). New objects entering the scene are especially problematic when pretraining for objectness, since they were never used as negative training examples and are thus assigned a high probability (see Fig. 1 (right) for an example).

The basic idea of our online adaptation scheme is to use pixels with very confident predictions as training examples. We select the pixels for which the predicted foreground probability exceeds a certain threshold α as positive examples. One could argue that using these pixels as positive examples is useless, since the network already gives very confident predictions for them. However, it is important that the adaptation retains a memory of the positive class in order to create a counterweight to the many negative examples being added. In our experiments, leaving out this step resulted in holes in the foreground mask.

We initially selected negative training examples in the same way, *i.e.* using pixels with a very low foreground probability. However, this led to degraded performance, probably, because during large appearance changes, false negative pixels will be selected as negative training examples, effectively destroying all chances to adapt to these changes. We thus select negative training examples in a different way, based on the assumption that the movement between two frames is small. The idea is to select all pixels which are very far away from the last predicted object mask. In order to deal with noise, the last mask can first be shrunk by an erosion operation. For our experiments, we use a square structural element with size 15, but we found that the exact value of this parameter is not critical. Afterwards, we compute a distance transform, which for each pixel provides the Euclidean distance to the closest foreground pixel of the mask. Finally, we apply a threshold d and treat all pixels with a distance larger than d as negative examples.

Pixels which are neither marked as positive nor as negative examples are assigned a "don't care" label and are ignored during the online updates. We can now fine-tune the network on the current frame, since every pixel has a label for training. However, in practice, we found that naively fine-tuning using the obtained training examples quickly leads to drift. To circumvent this problem, we propose to mix in the first frame as additional training examples during the online updates, since for the first frame the ground truth is available. We found that in order to obtain good results, the first frame should be sampled more often than the current frame, *i.e.* during online adaptation we perform a total of n_{online} update steps per frame, of which only n_{curr} are performed on the current frame, and the rest is performed on the first frame. Additionally, we reduce the weight of the loss for the current frame by a factor β (*e.g.* $\beta \approx 0.05$). A value of 0.05 might seem surprisingly small, but one has to keep in mind that the first frame is used very often for updates, quickly leading to smaller gradients, while the current frame is only selected a few times.

During online adaptation, the negative training examples are selected based on the mask of the preceding frame. Hence, it can happen that a pixel is selected as a negative example and that it is predicted as foreground at the same time. We call such pixels hard negatives. A common case in which hard negatives occur is when a previously unseen object enters the scene far away from the object of interest (see Fig. 1 (right)), which will then usually be detected as foreground by the network. We found it helpful to remove hard negatives from the foreground mask which is used in the next frame to determine negative training examples. This step allows selecting the hard negatives in the next frame again as negative examples. Additionally, we tried to adapt the network more strongly to hard negatives by increasing the number of update steps and/or the loss scale for the current frame in the presence of hard negatives. However, this did not improve the results further.

In addition to the previously described steps, we propose a simple heuristic which makes our method more robust against difficulties like occlusion: If (after the optional erosion) nothing is left of the last assumed foreground mask, we assume that the object of interest is lost and do not apply any online updates until the network again finds a non-empty foreground mask.

5 Experiments

Datasets. For objectness pretraining (*cf*. Section 3), we used the 1,464 training images of the PASCAL VOC 2012 dataset [11] plus the additional annotations provided by Hariharan *et al*. [16], leading to a total of 10,582 training images with 20 classes, which we all mapped to a single foreground class. For video object segmentation (VOS), we conducted most experiments on the recently introduced DAVIS dataset [34], which consists of 50 short full-HD video sequences, from which 30 are taken for training and 20 for validation. Consistent with most prior work, we conduct all experiments on the subsampled version with a resolution of 854×480 pixels. In order to show that our method generalizes, we also conducted experiments on the YouTube-Objects [19, 37] dataset for VOS, consisting of 126 sequences.

Experimental Setup. We pretrain on PASCAL and DAVIS, for 10 epochs each. For the baseline one-shot approach, we found 50 update steps on the first frame with a learning rate of $3 \cdot 10^{-6}$ to work well. For simplicity, we used a mini-batch size of only one image. Since DAVIS only has a training and a validation set, we tuned all hyperparameters on the training set of 30 sequences using three-fold cross validation, *i.e*. 20 training sequences are used for training and 10 for validation for each fold. As is standard practice, we augmented the training data by random flipping, scaling with a factor uniformly sampled from $[0.7, 1.3]$, and gamma augmentations [36].

For evaluation, we used the Jaccard index, *i.e*. the mean intersection-over-union (mIoU) between the predicted foreground masks and the ground truth masks. Results for additional evaluation measures suggested by Perazzi *et al*. [34] are shown in the supplementary material. We noticed that, especially for fine-tuning on the first frame, the random augmentations introduce non-negligible variations in the results. Hence, for these experiments, we conducted three runs and report mean and standard deviation values. All experiments were performed with our TensorFlow [1] based implementation, which we will make available together with pretrained models at https://www.vision.rwth-aachen.de/software/OnAVOS.

5.1 Baseline Systems

Effect of Pretraining Steps. Starting from the base network (*cf*. Section 3) our full baseline system (*i.e*. without adaptation) includes a first pretraining step on PASCAL for objectness, then on the training sequences of DAVIS, and finally a one-shot fine-tuning on the first frame. Each of these three steps can be enabled or disabled individually. Table 1 shows the results on DAVIS for all resulting combinations. As can be seen, each of these steps is useful since removing any step always deteriorates the results.

The base network was trained for a different task than binary segmentation and thus a new output layer needs to be learned at the same time as fine-tuning the rest of the network. Without pretraining on either PASCAL or DAVIS, the randomly initialized output layer is learned only from the first frame of the target sequence, which leads to a largely degraded performance of only 65.2% mIoU. However, when either PASCAL or DAVIS is used for pretraining, the result is greatly improved to 77.6% mIoU and 78.0% mIoU, respectively.

PASCAL	DAVIS	First frame	mIoU [%]
✓	✓	✓	**80.3** $\pm$ 0.4
	✓	✓	78.0 $\pm$ 0.1
✓		✓	77.6 $\pm$ 0.4
✓	✓		72.7
✓			65.3
	✓		71.0
		✓	65.2 $\pm$ 1.0

Table 1: Effect of (pre-)training steps on the DAVIS validation set. As can be seen, each of the three training steps are useful. The objectness pretraining step on PASCAL significantly improves the results.

While both results are very similar, it can be seen that PASCAL and DAVIS do provide complementary information, since using both datasets together further improves the result to 80.3%. We argue that the relatively large PASCAL dataset is useful for learning general objectness, while the limited amount of DAVIS data is useful to adapt to the characteristics (*e.g.* relatively high image quality) of the data of DAVIS, which provides an advantage for evaluating on DAVIS sequences.

Interestingly, even without looking at the segmentation mask of the first frame, *i.e.* in the unsupervised setup, we already obtain a result of 72.7% mIoU; slightly better than the current best unsupervised method FusionSeg [20], which obtains 70.7% mIoU on the DAVIS validation set[1] using objectness and optical flow as an additional cue.

Comparison to *OSVOS*. Without including their boundary snapping post-processing step, *OSVOS* achieves a result of 77.4% mIoU on DAVIS. Our system without objectness pre-training on PASCAL is directly comparable to this result and achieves 78.0% mIoU. We attribute this moderate improvement to the more recent network architecture which we adopted. Including PASCAL for objectness pretraining improves this result by further 2.3% to 80.3%.

5.2 Online Adaptation

Hyperparameter Study. As described in Section 4, *OnAVOS* involves relatively many hyperparameters. After some coarse manual tuning on the DAVIS training set, we found $\alpha = 0.97$, $\beta = 0.05$, $d = 220$, $n_{online} = 15$, $n_{curr} = 3$ to work well. While the initial 50 update steps on the first frame are performed with a learning rate of $3 \cdot 10^{-6}$, it proved useful to use a different learning rate $\lambda = 10^{-5}$ for the online updates on the current and the first frame. Starting from these values as the operating point, we conducted a more detailed study by changing one hyperparameter at a time, while keeping the others constant. We found that *OnAVOS* is not very sensitive to the choice of most hyperparameters and each configuration we tried performed better than the non-adapted baseline and we achieved only small improvements compared to the operating point (detailed plots are shown in the supplementary material). To avoid overfitting to the small DAVIS training set, we kept the values from the operating point for all further experiments.

[1]In FusionSeg [20], the result for all sequences including the training set is reported, but here we calculated the average only over the validation sequences for better comparability

Method	mIoU [%]
No adaptation	80.3 ± 0.4
Full adaptation	$\mathbf{82.8} \pm 0.5$
Only negatives	82.4 ± 0.3
Only positives	81.6 ± 0.3
No first frame during online adaptation	69.1 ± 0.2

Table 2: Online adaptation ablation experiments on the DAVIS validation set. As can be seen, mixing in the first frame during online updates is essential, and negative examples are more important than positive ones.

Method	DAVIS mIoU [%]	YouTube-Objects mIoU [%]
OnAVOS (ours), no adaptation	80.3 ± 0.4	76.1 ± 1.3
+CRF	81.7 ± 0.5	76.4 ± 0.2
+CRF +Test time augmentations	81.7 ± 0.2	76.6 ± 0.1
OnAVOS (ours), online adaptation	82.8 ± 0.5	76.8 ± 0.1
+CRF	84.3 ± 0.5	77.2 ± 0.2
+CRF +Test time augmentations	$\mathbf{85.7} \pm 0.6$	$\mathbf{77.4} \pm 0.2$
OSVOS [7]	79.8	72.5
MaskTrack [35]	79.7	72.6
LucidTracker [24] †	80.5	*76.2*
VPN [22]	75.0	-

Table 3: Comparison to the state of the art on the DAVIS validation set and the YouTube-Objects dataset. †: Concurrent work only published on arXiv. More results are shown in the supplementary material.

Ablation Study. Table 2 shows the results of the proposed online adaptation scheme and multiple variants, where parts of the algorithm are disabled, on the DAVIS validation set. Using the full method, we obtain an mIoU score of 82.8%. When disabling all adaptation steps, the performance significantly degrades to 80.3%, which demonstrates the effectiveness of the online adaptation method. The table further shows that negative training examples are more important than positive ones. If we do not mix in the first frame during online updates, the result is significantly degraded to 69.1% due to drift.

Timing Information. For the initial fine-tuning stage on the first frame, we used 50 update steps. Including the time for the forward pass for all further frames, this leads to a total runtime of around 90 seconds per sequence (corresponding to roughly 1.3 seconds per frame) of the DAVIS validation set using an NVIDIA Titan X (Pascal) GPU. When using online adaptation with $n_{online} = 15$, the runtime increases to around 15 minutes per sequence (corresponding to roughly 13 seconds per frame). However, our hyperparameter analysis revealed that this runtime can be significantly decreased by reducing n_{online} without much loss of accuracy. Note that for best results, *OSVOS* used a higher number of update steps on the first frame and needs about 10 minutes per sequence (corresponding to roughly 9 seconds per frame).

5.3 Comparison to State of the Art

Current state of the art methods use post-processing steps such as boundary snapping [7], or conditional random field (CRF) smoothing [24, 35] to improve the contours. In order to compare with them, we included per-frame post-processing using DenseCRF [26]. This might be especially useful since our network only provides one output for each 8×8 pixel block. Additionally, we added data augmentations during test time. To this end, we created 10 variants of each test image by random flipping, zooming, and gamma augmentations, and averaged the posterior probabilities over all 10 images.

In order to demonstrate the generalization ability of *OnAVOS* and since there is no separate training set for YouTube-Objects, we conducted our experiments on this dataset using the same hyperparameter values as for DAVIS, including the CRF parameters. Additionally, we omitted the pretraining step on DAVIS. Note that for YouTube-Objects, the evaluation protocols in prior publications sometimes differed by not including frames in which the object of interest is not present [24]. Here, we report results following the DAVIS evaluation protocol, *i.e.* including these frames, consistent with Khoreva *et al.* [24].

Table 3 shows the effect of our post-processing steps and compares our results on DAVIS and YouTube-Objects to other methods. Note that the effect of the test time augmentations is stronger when combined with online adaptation. We argue that this is because in this case, the augmentations do not only directly improve the end result as a post-processing step, but they also deliver better adaptation targets. On DAVIS, we achieve an mIoU of 85.7% which is, to the best of our knowledge significantly higher than any previously published result. Compared to *OSVOS*, this is an improvement of almost 6%. On YouTube-Objects, we achieve an mIoU of 77.4%, which is also a significant improvement over the second best result obtained by *LucidTracker* with 76.2%.

6 Conclusion

In this work, we have proposed *OnAVOS*, which builds on the *OSVOS* approach. We have demonstrated that the inclusion of an objectness pretraining step and our online adaptation scheme for semi-supervised video object segmentation are highly effective. We have further shown that our online adaptation scheme is robust against choices of hyperparameters and generalizes to another dataset. We expect that, in the future, more methods will adopt adaptation schemes which make them more robust against large changes in appearance. For future work, we plan to explicitly incorporate temporal context information into our method.

Acknowledgements. The work in this paper is funded by the EU project STRANDS (ICT-2011-600623) and the ERC Starting Grant project CV-SUPER (ERC-2012-StG-307432). We would like to thank István Sárándi, Jörg Stückler, Lucas Beyer, and Aljoša Ošep for helpful discussions and proofreading.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] M. Babaee, D. T. Dinh, and G. Rigoll. A deep convolutional neural network for background subtraction. *arXiv preprint arXiv:1702.01731*, 2017.

[3] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: Robust video object cutout using localized classifiers. *ACM Trans. Graphics*, 28(3):70:1–70:11, 2009.

[4] X. Bai, J. Wang, and G. Sapiro. Dynamic color flow: A motion-adaptive color model for object segmentation in video. In *ECCV*, 2010.

[5] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV Workshops*, 2016.

[6] M. Braham and M. Van Droogenbroeck. Deep background subtraction with scene-specific convolutional neural networks. In *Int. Conf. on Systems, Signals and Image Proc.*, 2016.

[7] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *CVPR*, 2017.

[8] J. Chang, D. Wei, and J. W. Fisher III. A video representation using temporal superpixels. In *CVPR*, 2013.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[10] L. F. Ellis and V. Zografos. Online learning for fast segmentation of moving objects. In *ACCV*, 2012.

[11] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge: A retrospective. *IJCV*, 111(1): 98–136, 2015.

[12] Q. Fan, F. Zhong, D. Lischinski, D. Cohen-Or, and B. Chen. Jumpcut: Non-successive mask transfer and interpolation for video cutout. *SIGGRAPH Asia*, 34(6), 2015.

[13] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Europ. Conf. on Comput. Learning Theory*, 1995.

[14] H. Grabner and H. Bischof. On-line boosting and vision. In *CVPR*, 2006.

[15] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010.

[16] B. Hariharan, P. Arbelaez, L. D. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016.

[19] S. D. Jain and K. Grauman. Supervoxel-consistent foreground propagation in video. In *ECCV*, 2014.

[20] S. D. Jain, B. Xiong, and K. Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. In *CVPR*, 2017.

[21] S. D. Jain, B. Xiong, and K. Grauman. Pixel objectness. *arXiv preprint arXiv:1701.05349*, 2017.

[22] V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. In *CVPR*, 2017.

[23] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *PAMI*, 34(7): 1409–1422, 2012.

[24] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele. Lucid data dreaming for object tracking. In *arXiv preprint arXiv: 1703.09554*, 2017.

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

[26] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*, 2011.

[27] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *Trans. Image Proc.*, 25(4):1834–1848, 2016.

[28] T-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[29] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[30] N. Maerki, F. Perazzi, O. Wang, and A. Sorkine-Hornung. Bilateral space video segmentation. In *CVPR*, 2016.

[31] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

[32] K. E. Papoutsakis and A. A. Argyros. Integrating tracking with fine object segmentation. *Image and Vision Computing*, 31(10):771–785, 2013.

[33] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung. Fully connected object proposals for video segmentation. In *ICCV*, 2015.

[34] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[35] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung. Learning video object segmentation from static images. In *CVPR*, 2017.

[36] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *CVPR*, 2017.

[37] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, 2012.

[38] S. A. Ramakanth and R. V. Babu. Seamseg: Video object segmentation using patch seams. In *CVPR*, 2014.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[40] P. Tokmakov, K. Alahari, and C. Schmid. Learning motion patterns in videos. In *CVPR*, 2017.

[41] Y-H. Tsai, M-H. Yang, and M. J. Black. Video segmentation via object flow. In *CVPR*, 2016.

[42] Wenguan W. and Shenjian B. Super-trajectory for video segmentation. *arXiv preprint arXiv:1702.08634*, 2017.

[43] N. Wang and D-Y. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013.

[44] Y. Wang, Z. Luo, and P.-M. Jodoin. Interactive deep learning method for segmenting moving objects. *Pattern Recognition Letters*, 2016.

[45] Z. Wu, C. Shen, and A. v. d. Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*, 2016.

[46] Z. Wu, C. Shen, and A. v. d. Hengel. Bridging category-level and instance-level semantic image segmentation. *arXiv preprint arXiv:1605.06885*, 2016.

[47] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

Supplementary Material

A More Comprehensive Comparison to Other Methods

Table 4 shows a more comprehensive comparison of our results to the results obtained by other methods.

Method	DAVIS mIoU [%]	YouTube-Objects mIoU [%]
OnAVOS (ours), no adaptation	81.7 ± 0.2	76.6 ± 0.1
OnAVOS (ours), online adaptation	$\mathbf{85.7} \pm 0.6$	$\mathbf{77.4} \pm 0.2$
OSVOS [7]	79.8	72.5
MaskTrack [35]	79.7	72.6
LucidTracker [24] †	80.5	76.2
VPN [22]	75.0	-
FCP [33]	63.1	-
BVS [30]	66.5	59.7
OFL [41]	71.1	70.1
STV [42]	73.6	-

Table 4: Comparison to other methods on the DAVIS validation set and the YouTube-Objects dataset. Note that *MaskTrack* [35] and *LucidTracker* [24] report results on DAVIS for all sequences including the training set, but here we show their results for the validation set only. †: Concurrent work only published on arXiv.

B Additional Evaluation Measures for DAVIS

Table 5 shows a more detailed evaluation on the DAVIS validation set using the evaluation measures suggested by Perazzi *et al.* [34]. The measures used here are the Jaccard index $\mathcal{J}$, defined as the mean intersection-over-union (mIoU) between the predicted foreground masks and the ground truth masks; the contour accuracy measure $\mathcal{F}$, which measures how well the segmentation boundaries agree; and the temporal stability measure $\mathcal{T}$, which measures the consistency of the predicted masks over time. For more details of these measures, we refer the interested reader to Perazzi *et al.* [34]. Note that the results for additional measures for *LucidTracker* [24] are missing since they are only reported averaged over all 50 sequences of DAVIS and not on the validation set.

The table shows that each evaluation measure is significantly improved by the proposed online adaptation scheme. *OnAVOS* obtains the best mean results for all three measures. It is surprising that our result for the temporal stability $\mathcal{T}$ is better than the result by *MaskTrack* [35], although in contrast to our method, they explicitly incorporate temporal context by propagating masks.

C Per-Sequence Results for DAVIS

Table 6 shows mIoU results for each of the 20 sequences of the DAVIS validation set. On 18 out of 20 sequences, *OnAVOS* obtains either the best or the second best result.

Measure		*OnAVOS* (ours)		*OSVOS* [7]	*MaskTrack* [35]	*LucidTracker* [24]
		Un-adapted	Adapted			
$\mathcal{J}$	mean ↑	*81.7* ± 0.2	**85.7** ± 0.6	79.8	79.7	80.5
	recall ↑	92.2 ± 0.6	**95.4** ± 0.8	*93.6*	93.1	-
	decay ↓	11.9 ± 0.3	**7.1** ± 1.7	14.9	*8.9*	-
$\mathcal{F}$	mean ↑	*81.1* ± 0.2	**84.2** ± 0.8	80.6	75.4	-
	recall ↑	88.2 ± 0.3	*88.7* ± 1.3	**92.6**	87.1	-
	decay ↓	11.2 ± 0.5	**7.8** ± 1.8	15.0	*9.0*	-
$\mathcal{T}$	mean ↓	27.3 ± 2.2	**18.5** ± 0.1	37.6	*21.8*	-

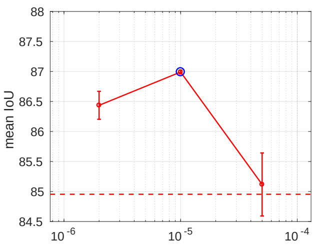
Table 5: Additional evaluation measures on the DAVIS validation set. Best and second best results are highlighted with bold and italic fonts, respectively.

Sequence	Method, mIoU [%]				
	OnAVOS (ours)		*OSVOS* [7]	*MaskTrack* [35]	*LucidTracker* [24]
	Un-adapted	Adapted			
blackswan	*96.1* ± 0.1	**96.2** ± 0.1	94.2	90.3	95.0
bmx-trees	48.2 ± 0.8	*57.0* ± 1.0	55.5	**57.5**	55.0
breakdance	62.6 ± 4.2	73.6 ± 3.8	70.8	*76.1*	**87.2**
camel	84.6 ± 0.1	*85.5* ± 0.1	85.1	80.1	**94.3**
car-roundabout	86.5 ± 0.2	**97.5** ± 0.0	95.3	*96.0*	*96.0*
car-shadow	*94.1* ± 0.1	**96.8** ± 0.1	93.7	93.5	90.3
cows	**95.4** ± 0.0	**95.4** ± 0.0	*94.6*	88.2	93.1
dance-twirl	78.4 ± 0.7	*85.6* ± 1.0	67.0	84.4	**88.6**
dog	**95.6** ± 0.1	**95.6** ± 0.1	90.7	90.8	*95.0*
drift-chicane	*87.4* ± 0.5	**89.2** ± 0.2	83.5	86.2	1.4
drift-straight	*81.3* ± 5.6	**93.7** ± 0.9	67.6	56.0	79.9
goat	*90.8* ± 0.1	**91.4** ± 0.1	88.0	84.5	88.9
horsejump-high	*89.3* ± 0.3	**90.1** ± 0.0	78.0	81.8	87.1
kite-surf	**70.1** ± 1.0	*69.1* ± 0.1	68.6	60.0	64.6
libby	*87.1* ± 1.0	**88.6** ± 0.1	80.8	77.5	85.5
motocross-jump	**89.7** ± 0.2	70.4 ± 11.9	*81.6*	68.3	75.1
paragliding-launch	**64.6** ± 0.1	*64.3* ± 0.1	62.5	62.1	63.7
parkour	92.4 ± 0.2	**93.6** ± 0.0	85.6	88.2	*93.2*
scooter-black	64.8 ± 7.1	**91.3** ± 0.1	71.1	82.4	*86.5*
soapbox	74.0 ± 4.6	89.8 ± 1.2	81.2	*89.9*	**90.5**
mean	*81.7* ± 0.2	**85.7** ± 0.6	79.8	79.7	80.5

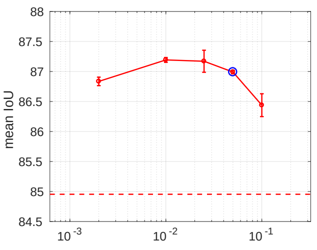
Table 6: Per-sequence results on the DAVIS validation set. Best and second best results are highlighted with bold and italic fonts, respectively.

D Hyperparameter Study on DAVIS

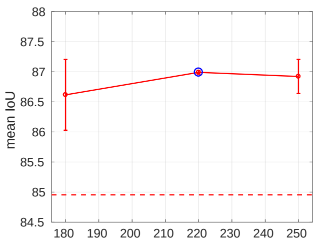
As described in the main paper, we found $\alpha = 0.97$, $\beta = 0.05$, $d = 220$, $n_{online} = 15$, $n_{curr} = 3$, $\lambda = 10^{-5}$ and 15 for the erosion size to work well on DAVIS. Starting from these values as the operating point, we conducted a more detailed hyperparameter study by changing one hyperparameter at a time, while keeping all others constant (see Fig. 3). The plots show that the performance of *OnAVOS* is in general very stable with respect to the choice of most of its hyperparameters and for every configuration we tried, the result was better than the un-adapted baseline (the dashed line in the plots). The single most important hyperparameter is the online learning rate λ, which is common for deep learning approaches. The online loss scale β and the positive threshold α have a moderate influence on performance, while changing the distance threshold d and the number of steps n_{online} and n_{curr} in a reasonable range only leads to minor changes in accuracy. For the erosion



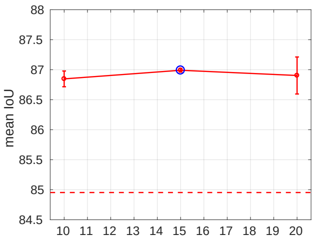
(a) online learning rate λ



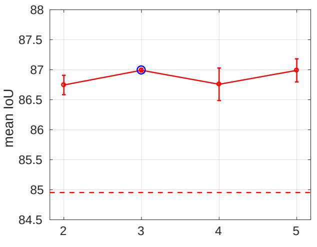
(b) online loss scale β



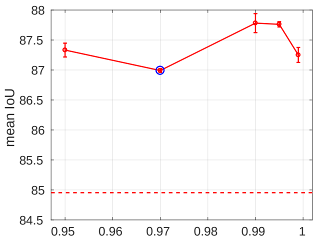
(c) distance threshold d



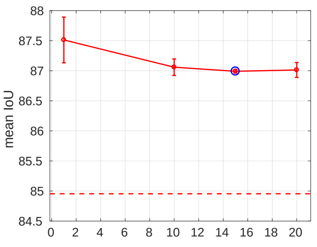
(d) total steps n_{online}



(e) update steps n_{curr}



(f) positive threshold α



(g) erosion size

Figure 3: Influence of online adaptation hyperparameters on the DAVIS training set. The blue circle marks the operating point, based on which one parameter is changed at a time. The dashed line marks the un-adapted baseline. The plots show that overall our method is very robust against the exact choice of hyperparameters, except for the online learning rate λ. The standard deviations estimated by three runs are shown as error bars. In some cases, including the operating point, the estimated standard deviation is so small that it is hardly visible.

size, the optimum is achieved at 1, *i.e.* when no erosion is applied. This result suggests that the erosion operation is not helpful for DAVIS. The plots show that there is still some potential for improving the results by further tuning the hyperparameters. However, this study was meant as a characterization of our method rather than a systematic tuning.

The generalizability and the robustness of *OnAVOS* with respect to the choice of hyperparameters is further confirmed by the experiments on YouTube-Objects, which used the same hyperparameter settings as on DAVIS.