Learning Fast and Robust Target Models for Video Object Segmentation

Andreas Robinson\textsuperscript{1*} \hspace{1cm} Felix Järemo Lawin\textsuperscript{1*} \hspace{1cm} Martin Danelljan\textsuperscript{2}
Fahad Shahbaz Khan\textsuperscript{1,3} \hspace{1cm} Michael Felsberg\textsuperscript{1}
\textsuperscript{1}CVL, Linköping University, Sweden \hspace{1cm} \textsuperscript{2}CVL, ETH Zurich, Switzerland \hspace{1cm} \textsuperscript{3}IIAI, UAE

Abstract

Video object segmentation (VOS) is a highly challenging problem since the initial mask, defining the target object, is only given at test-time. The main difficulty is to effectively handle appearance changes and similar background objects, while maintaining accurate segmentation. Most previous approaches fine-tune segmentation networks on the first frame, resulting in impractical frame-rates and risk of overfitting. More recent methods integrate generative target appearance models, but either achieve limited robustness or require large amounts of training data.

We propose a novel VOS architecture consisting of two network components. The target appearance model consists of a light-weight module, which is learned during the inference stage using fast optimization techniques to predict a coarse but robust target segmentation. The segmentation model is exclusively trained offline, designed to process the coarse scores into high quality segmentation masks. Our method is fast, easily trainable and remains highly effective in cases of limited training data. We perform extensive experiments on the challenging YouTube-VOS and DAVIS datasets. Our network achieves favorable performance, while operating at higher frame-rates compared to state-of-the-art. Code and trained models are available at \url{https://github.com/andr345/frtm-vos}.

1. Introduction

The problem of video object segmentation (VOS) has a variety of important applications, including object boundary estimation for grasping [1, 25], autonomous driving [38, 40], surveillance [10, 13] and video editing [33]. The task is to predict pixel-accurate masks of the region occupied by a specific target object, in every frame of a given video sequence. This work focuses on the semi-supervised setting, where a target ground-truth mask is provided in the first frame. Challenges arise in dynamic environments with similar background objects and when the target undergoes considerable appearance changes or occlusions. Successful video object segmentation therefore requires both robust and accurate target pixel classification.

Aiming to achieve a robust target-specific segmentation, several methods [4, 30, 36, 47] fine-tune a generic segmentation network on the first frame, given the ground-truth mask. Although capable of generating accurate segmentation masks under favorable circumstances, these methods suffer from low frame-rates, impractical for many real world applications. Moreover, fine-tuning is prone to overfit to a single view of the scene, while degrading generic segmentation functionality learned during offline training. This limits performance in more challenging videos involving drastic appearance changes, occlusions and distractor objects [48]. Further, the crucial fine-tuning step is not included in the offline training stage, which therefore does not simulate the full inference procedure.

Recent works [20, 22, 33, 34, 44] address these limitations by employing internal models of the target and back-
ground appearance. They are based on, e.g., feature concatenation [33], feature matching [20, 34, 44] or Gaussian models [22]. Such generative models have the advantage of facilitating efficient closed-form solutions that are easily integrated into neural networks. A drawback to these methods is the demand for large amounts of data in order to learn representations applicable for the internal models [33, 34, 46]. Due to the limited availability of annotated video data, these methods rely heavily on pre-training on image segmentation and synthesized VOS data via augmentation techniques. On the other hand, discriminative methods generally yield superior predictive power [32] and have thus been preferred in many vision tasks, including image recognition [39], object detection [28] and tracking [26]. In this work, we therefore tackle the problem of integrating a discriminative model of the target appearance into a VOS architecture.

Our approach integrates a light-weight discriminative target model and a segmentation network, for modeling the target appearance and generating accurate segmentation masks. Operating on deep features, the proposed target model learns during inference to provide robust segmentation scores. The segmentation network is designed to process features with the segmentation scores as guidance. During offline training the network learns to accurately adhere to object edges and to suppress erroneous classification scores from the target model, see Figure 1. To learn the network parameters, we propose a training strategy that simulates the inference stage. This is realized by optimizing the target model on reference frames in each batch, and back-propagating the segmentation errors on corresponding validation frames. Contrary to fine-tuning based methods, the target adaption process is thus fully simulated during the offline training stage. During inference we keep the segmentation network fixed, while the target-specific learning is entirely performed by the target appearance model. Consequently, the segmentation network is target agnostic, retaining generic object segmentation functionality.

Unlike previous state-of-the-art methods, our discriminative target model requires no pre-training for image and synthetic video segmentation data. Our final approach, consisting of a single network architecture, is trained on VOS data in a single phase. Further, the employment of Gauss-Newton based optimization enables real-time video segmentation. We perform experiments on the DAVIS [37] and YouTube-VOS 2018 [48] datasets and demonstrate the impact of the components of our proposed approach in an ablative analysis. We further compare our approach to several state-of-the-art methods. Despite its simplicity, our approach achieves an overall score of 76.7 on DAVIS 2017 and 72.1 on YouTube-VOS, while operating at 22 frames per second (FPS). We also evaluate a faster version of our approach that achieves a speed of 41 FPS, with only a slight degradation in segmentation accuracy.

2. Related work

The task of video object segmentation has seen extensive study and rapid development in recent years, largely driven by the introduction and evolution of benchmarks such as DAVIS [37] and YouTube-VOS [48].

First-frame fine-tuning: Most state-of-the-art approaches train a segmentation network offline, and then fine-tune it on the first frame [4, 30, 36, 47] to learn the target-specific appearance. This philosophy was extended [43] by additionally fine-tuning on subsequent video frames. Other approaches [8, 19, 29] further integrate optical flow as an additional cue. While obtaining impressive results on the DAVIS 2016 dataset, the extensive fine-tuning leads to impractically long run-times. Furthermore, such extensive fine-tuning is prone to overfitting, a problem only partially addressed by heavy data augmentation [23].

Non-causal methods: Another line of research approaches the VOS problem by allowing non-causal processing [2, 9, 21, 27]. In this work, we focus on the causal setting in order to accommodate real-time applications.

Mask propagation: Several recent methods [22, 33, 34, 36, 46, 49] employ a mask-propagation module to improve spatio-temporal consistency of the segmentation. In [36], the model is learned offline to predict the target mask through refinement of the previous frame’s segmentation output. To further avoid first-frame fine-tuning, some approaches [33, 49] concatenate the current frame features with the previous mask and a target representation generated in the first frame. Unlike these methods, we do not explicitly enforce spatio-temporal consistency through mask-propagation. Instead, we use previous segmentation masks as training data for the discriminative model.

Feature matching: Recent methods [6, 20, 33, 34, 44, 45, 46] incorporate feature matching to locate the target object. Rather than fine-tuning the network on the first frame, these methods first construct appearance models from features corresponding to the initial target labels. Features from incoming frames are then classified using techniques inspired by classical clustering methods [6, 22] or feature matching [20, 44, 46]. In [34], a dynamic memory is used to combine feature matching from multiple previous frames.

Tracking: Efficient online learning of discriminative target-specific appearance models has been explored in visual tracking [15, 17]. Recently, optimization-based trackers [3, 11, 12] have achieved impressive results on benchmarks. These methods train convolution filters using efficient optimization to discriminate between target and background. The close relation between the two problem domains is made explicit in [7], where object trackers are used as external components to locate the target. Gauss-Newton has previously been used in object segmentation [41] for pose estimation of known object shapes. In contrast, we do not employ off-the-shelf trackers to predict the target or rely
on target pose estimation. Instead we take inspiration from the optimization-based learning of a discriminative model, in order to capture the target object appearance.

3. Method

In this work, we tackle the problem of predicting accurate segmentation masks of a target object, defined in the first frame of the video. This is addressed by constructing two network modules, \( D \) and \( S \), specifically designed for target modeling and segmentation respectively. The target model \( D(x; w) \) takes features \( x \) as input and generates a coarse, but robust, segmentation output \( s = D(x; w) \) of the target object. It is parametrized by the weights \( w \), which are solely learned during inference using the first-frame ground-truth, in order to capture the appearance of the target object.

The coarse segmentation scores \( s \), generated by the target model \( D \), is passed to the segmentation network \( S(s, x; \theta) \), additionally taking backbone features \( x = F(I) \). The parameters \( \theta \) of the segmentation network is only trained during the offline training stage to predict the final high-resolution segmentation of the target. The coarse segmentation \( s \) thus serves as a robust guide, indicating the target location. Crucially, this allows the segmentation network to remain target agnostic, and learn generic segmentation functionality. Since \( S \) is trained with coarse segmentation inputs \( s \) generated by the target model, it learns to enhance its prediction and correct mistakes.

During inference, we update the target model using the segmentation masks generated by \( S \). Specifically, the mask and associated features are stored in a memory \( \mathcal{M} \). Before the next incoming frame, we further adapt our model to the target appearance by re-optimizing \( D \) over all samples in \( \mathcal{M} \). In contrast to simply re-training on the latest frame, adding more training data to \( \mathcal{M} \) over time, reduces the risk for model drifting. Our full VOS architecture is illustrated in Figure 2.

3.1. Target model

We aim to develop a powerful and discriminative target appearance model, capable of differentiating between the target and background image regions. To successfully accommodate the VOS problem, the model must be robust to appearance changes and distractor objects. Moreover, it needs to be easily updated with new data and efficiently trainable. To this end, we employ a light-weight linear model \( D(x; w) \) realized as two convolutional layers,

\[
s = D(x; w) = w_2 \ast (w_1 \ast x),
\]

with parameters \( w \). These are trained exclusively during inference with image features \( x \) and the target segmentation mask \( y \) given in the first video frame. It then takes input feature maps \( x \) from subsequent video frames and outputs coarse segmentation scores \( s \). The factorized formulation (1) is used for efficiency, where the first layer \( w_1 \) reduces the feature dimensionality and the second layer \( w_2 \) computes the actual segmentation scores.

Fundamental to our approach, the target model parameters \( w \) must be learned with minimal computational impact. To enable the deployment of fast converging optimization techniques, we adopt an \( L^2 \) loss given by,

\[
L_D(w; \mathcal{M}) = \sum_k \gamma_k \| y_k - U(D(x_k)) \|^2 + \sum_j \lambda_j \| w_j \|^2.
\]

Here, the parameters \( \lambda_j \) control the regularization term and \( v_k \) are weight masks balancing the impact of target and background pixels. \( U \) denotes bilinear up-sampling of the output from the target model to the spatial resolution of the labels \( y_k \). The memory \( \mathcal{M} = \left\{(x_k, y_k, \gamma_k)\right\}_{k=1}^K \) consists of sample feature maps \( x_k \), target labels \( y_k \) and sample weights \( \gamma_k \). During inference, \( \mathcal{M} \) is updated with new samples from the video sequence. To add more variety in the initial frame, we generate additional augmented samples \( \{\tilde{x}_k, \tilde{y}_k, \tilde{\gamma}_k\} \) using the initial image \( I_0 \) and labels \( y_0 \), see supplement for more details. Compared to blindly updating on the latest frame \([43]\), \( \mathcal{M} \) provides a controlled means of adding new samples while keeping past frames in memory by setting appropriate sample weights \( \gamma_k \).

Optimization: We employ the Gauss-Newton (GN) based strategy from [11] to optimize the parameters \( w \). In comparison to the commonly used gradient descent based approaches, this strategy has significantly faster convergence properties [31]. In each iteration, the optimal increment...
Δw is found using a quadratic approximation of the loss in (2)
\[ L_D(w + \Delta w) \approx \Delta w^T J_w^T J_w \Delta w + 2 \Delta w^T J_w^T r_w + r_w^T r_w. \]

Here, \( r_w \) contains the residuals (2) as \( \sqrt{\sum_k v_k \cdot (y_k - U(D(x_k)))} \) and \( \sqrt{\sum_j w_j \cdot J_w} \) and \( J_w \) is the Jacobian of the residuals \( r_w \) at \( w \) and. The objective (3) results in a positive definite quadratic problem, which we minimize over \( \Delta w \) with Conjugate Gradient (CG) descent [18]. We then update \( w \leftarrow w + \Delta w \) and execute the next GN iteration.

**Pixel weighting:** To address the imbalance between target and background, we employ a weight mask \( v \) in (2) to ensure that the target influence is not too small relative to the usually much larger background region. We define the target influence as the fraction of target pixels in the image \( \hat{\kappa}_k = N^{-1} \sum_n y_k(n) \), where \( n \) is the pixel index and \( N \) the total number of pixels. The weight mask is then defined as
\[ v_k = \begin{cases} \kappa/\hat{\kappa}_k, & (y_k)_n = 1 \\ (1 - \kappa)/(1 - \hat{\kappa}_k), & (y_k)_n = 0 \end{cases} \]

where \( \kappa = \max(\kappa_{\min}, \hat{\kappa}_k) \) is the desired and \( \hat{\kappa}_k \) the actual target influence. We set \( \kappa_{\min} = 0.1 \) in our approach.

### 3.2. Segmentation Network

While the target model provides robust but coarse segmentation scores, the final aim is to generate an accurate segmentation mask of the target at the original image resolution. To this end, we introduce a segmentation network, that processes the coarse score \( s \) along with backbone features. The network consists of two types of building blocks: a target segmentation encoder (TSE) and a refinement module (see Figure 3). From these we construct a U-Net based architecture for object segmentation as in [50]. Unlike most state-of-the-art methods for semantic segmentation [5, 51], the U-Net structure does not rely on dilated convolutions, but effectively integrates low-resolution deep feature maps. This is crucial for reducing the computational complexity of our target model during inference.

The segmentation network takes features maps \( x^d \) as input from multiple depths in the backbone feature extractor network, with decreased resolution at each depth \( d \). For each layer, \( x^d \) along with the coarse scores \( s \) are first processed by a TSE block \( T^d \). The refinement module \( R^d \) then inputs the resulting segmentation encoding generated by \( T^d \) and the refined outputs \( z^{d+1} \) from the preceding deeper layer \( z^d = R^d(T^d(x^d, s), z^{d+1}) \). The refinement modules are comprised of two residual blocks and a channel attention block (CAB), as in [50]. For the deepest block we set \( z^{d+1} \) to an intermediate projection of \( x^d \) inside \( T^d \). The output \( z^1 \) at the shallowest layer is processed by two convolutional layers, providing the final refined segmentation output \( \hat{y} \).

**Target segmentation encoder:** Seeking to integrate features and scores, we introduce the target segmentation encoder (TSE). It processes features in two steps, as visualized in Figure 3 (right). First, we project the backbone features to 64 channels to reduce the subsequent computational complexity. We maintain 64 channels throughout the segmentation network, keeping the number of parameters low. After projection, the features are concatenated with the segmentation score \( s \) and encoded by three convolutional layers.

### 3.3. Offline Training

We learn the parameters in our segmentation network offline by training on VOS training data. To this end, we propose a training scheme to simulate the inference stage. The network is trained on samples consisting of one reference frame and one or more validation frames. These are all randomly selected from the same video sequence. A training iteration is then performed as follows: We first optimize the target model weights \( w \), described in Section 3.1, based on the reference frame. We then apply our full network, along with the learned target model, on the validation frames to predict the target segmentations. The parameters in the network are learned by back-propagating through the binary cross-entropy loss with respect to the ground-truth masks.

During offline training, we only learn the parameters of the segmentation network, and freeze the weights of the feature extractor. Since the target model only receives backbone features, we can pre-learn and store the target model weights for each sequence. The offline training time is therefore not significantly affected by the learning of \( D \).

The network is trained in a single phase on VOS data. We select one reference frame and two validation frames per sample and train the segmentation network with the ADAM optimizer [24]. We start with the learning rate \( \alpha = 10^{-3} \), moment decay rates \( \beta_1 = 0.9, \beta_2 = 0.999 \) and weight decay \( 10^{-5} \), and train for about 10^6 iterations, split into 120 epochs. The learning rate is then reduced to \( \alpha = 10^{-4} \), and we train for another 60 epochs. With pre-learned target model weights, the training is completed in less than a day.
For efficiency, we keep the first layer of the target model in the sequence are prioritized in the re-optimization of the model. On the other hand, it is important that the target model is updated frequently, for objects that undergo rapid appearance changes. In our approach we set the augmented dataset. Each new sample is given a weight \( \gamma_i = (1 - \eta)\gamma_{i-1} \), with \( \gamma_0 = \eta \). We then normalize the sample weights to sum to unity. The parameter \( \eta < 1 \) controls the update rate, such that the most recent samples in the sequence are prioritized in the re-optimization of the target model \( D \). For practical purposes, we limit the maximum capacity \( K_{\text{max}} \) of the dataset. When the maximum capacity is reached, we remove the sample with the smallest weight from \( M_{i-1} \) before inserting a new one. In all our experiments we set \( \eta = 0.1 \) and \( K_{\text{max}} = 80 \).

During inference, we optimize the target model parameters \( w_1 \) and \( w_2 \) on the current dataset \( M_i \) every \( t_s \)-th frame. For efficiency, we keep the first layer of the target model \( w_1 \) fixed during updates. Setting \( t_s \) to a large value reduces the inference time and regularizes the update of the target model. On the other hand, it is important that the target model is updated frequently, for objects that undergo rapid appearance changes. In our approach we set \( t_s = 8 \). The framework supports multi object segmentation by employing a target model for each object and fuse the final refined predictions with softmax aggregation as in [33]. We only require one feature extraction per image, since the features \( x_i \) are common for all target objects.

### 3.4. Inference

**Algorithm 1 Inference**

**Input:** Images \( I_1 \), target \( y_0 \)

1. \( M_0(I_0, y_0) = \{(x_k, \hat{y}_k, \gamma_k)\}_{k=1}^{K} \) # Init dataset, sec 3.1
2. \( w_0 = \text{optimize}(L(D(w; M_0))) \) # Init \( D \), sec 3.1
3. for \( i = 1, 2, \ldots \) do
4. \( x_i = F(I_i) \) # Extract features
5. \( s_i = D(x_i; w_{i-1}) \) # Predict target, sec 3.1
6. \( \hat{y}_i = S(x_i; s_i) \) # Segment target, sec 3.2
7. \( M_i = \text{extend}(x_i, \hat{y}_i; \gamma_i; M_{i-1}) \) # Extend dataset
8. if \( i \mod t_s = 0 \) then # Update \( D \) every \( t_s \) frame
9. \( w_i = \text{optimize}(L(D(w_i; M_i))) \)

### 3.5. Implementation details

We implement our method in the PyTorch framework [35] and use a ResNet [16], pre-trained on ImageNet [39], as the feature extractor \( F \). Following the naming convention in Table 1 of [16], we extract four feature maps from the outputs of the blocks \( \text{conv}2_x \) through \( \text{conv}5_x \). The target model \( D \) accepts features from \( \text{conv}4_x \) and produces 1-channel score maps. Both the input features and output scores have a spatial resolution 1/16th of the input image.

**Target model:** The first layer \( w_1 \) has \( 1 \times 1 \) kernels reducing input features to \( e = 96 \) channels while \( w_2 \) has a \( 3 \times 3 \) kernel with one output channel. During first-frame optimization, \( w_1 \) and \( w_2 \) are randomly initialized. Using the data augmentation (see the supplementary material), we generate a initial dataset \( M_0 \) of 5 image and label pairs. We then optimize \( w_1 \) and \( w_2 \) with the Gauss-Newton algorithm outlined in Section 3.1 with \( N_{\text{GN}} = 5 \) GN steps. We apply \( N_{\text{CG}} = 10 \) CG iterations in all GN steps but the first one. Since the initialization is random, we reduce the number of iterations to \( N_{\text{CG}} = 5 \) in the first step. In the target model update step we use \( N_{\text{CG}} = 10 \) CG iterations, updating \( w_2 \) every \( t_s = 8 \) frame, while keeping \( w_1 \) fixed. We employ the aforementioned settings with a ResNet-101 backbone in our final approach, denoted Ours in the following sections.

We additionally develop a fast version, named Ours-fast, with a ResNet-18 backbone and fewer optimization steps. Specifically, we set \( N_{\text{GN}} = 4 \), \( N_{\text{CG}} = 5 \), \( N_{\text{CG}} = 10 \) and \( N_{\text{CG}} = 5 \) in all GN steps.

### 4. Experiments

We perform experiments on three benchmarks: DAVIS 2016 [37], DAVIS 2017 [37] and YouTube-VOS [48]. For YouTube-VOS, we compare on the official validation set, with withheld ground-truth. For ablative experiments, we also show results on a separate validation split of the YouTube-VOS train set, consisting of 300 videos not used for training. Following the standard DAVIS protocol, we report both the mean Jaccard \( J \) index and mean boundary \( F \) scores, along with the overall score \( J \& F \), which is the mean of the two. For comparisons on YouTube-VOS, we report \( J \) and \( F \) scores for classes included in the training set (seen) and the ones that are not (unseen). The overall score \( G \) is computed as the average over all four scores, defined in YouTube-VOS. In addition, we compare the computational speed of the methods in terms of frames per second (FPS), computed by taking the average over the DAVIS 2016 validation set. For our approach, we used a V100 GPU and included all steps in Algorithm 1 to compute the frame rates. Further results and analysis are provided in the supplement.

#### 4.1. Ablation study

We analyze the contribution of the key components in our approach. All compared approaches are trained using the YouTube-VOS training split.

**Base net:** We construct a baseline network to analyze the impact of our target model \( D \). This is performed by replacing \( D \) with an offline-trained target encoder, and retraining
the segmentation network $S$. As for our proposed network we keep the backbone $F$ parameters fixed. The target encoder is comprised of two convolutional layers, taking reference frame features from ResNet blocks $\text{conv4}_x$ and the corresponding target mask as input. Features ($\text{conv4}_x$) extracted from the test frame are concatenated with the output from the target encoder and processed with two additional convolutional layers. The output is then passed to the segmentation network $S$ in the same manner as for the coarse segmentation score $s$ (see Section 3.2). We train this model with the same methodology as for our network.

**E-T:** We integrate a first-frame fine-tuning strategy into our network to compare this to our discriminative target model. For this purpose, we create an initial dataset $M_0$ with 20 samples using the same sample generation procedure employed for our approach (section 3.1). We then fine-tune all components of the network, except for the feature extractor, with supervision on the target model (loss in (2)) and the pre-trained segmentation network (binary cross-entropy loss) using the ADAM optimizer with 100 iterations and a batch size of four. In this setting we omitted the proposed optimization strategy of the target model and instead initialized the parameters randomly before fine-tuning.

**D-only - no update:** To analyze the impact of the segmentation network $S$, we remove it from our architecture and instead let the target-specific model $D$ output the final segmentations. The coarse target model predictions are upsampled to full image resolution through bilinear interpolation. In this version, we only train the target model $D$ on the first frame, and refrain from subsequent updates.

**D-only:** We further enable target model updates (as described in Section 3.4) using the raw target predictions.

**Ours - no update:** For a fair comparison, we evaluate a variant of our approach with the segmentation network, but without any update of the target model $D$ during inference.

**Ours:** Finally, we include target model updates with segmentation network predictions to obtain our final approach.

In Table 1, we present the results in terms of the $\mathcal{J}$ score on a separate validation split of the YouTube-VOS training dataset. The base network, not employing the target model $D$, achieves a score of 49.8%. Employing fine-tuning on the first frame leads to an absolute improvement of 6%. Remarkably, using only the linear target model $D$ is on par with online fine-tuning. While fine-tuning an entire segmentation network is prone to severe overfitting to the initial frame, our shallow target model has limited capacity, acting as an implicit regularization mechanism that benefits robustness and generalization to unseen aspects of the target and background appearance. Including updates results in an absolute improvement of 1.6%, demonstrates that we benefit from online updates despite the coarseness of the target mode generated labels. Further adding the segmentation network $S$ (Ours - no update) leads to a major absolute gain of 8.3%. This improvement stems from the offline-learned processing of the coarse segmentations, yielding more accurate mask predictions. Finally, the proposed online updating strategy additionally improves the score to 71.4%.

### 4.2. Comparison to state-of-the-art

We compare our method to recent approaches on the YouTube-VOS, DAVIS 2017 and DAVIS 2016 benchmarks. We provide results for two versions of our approach: **Ours** and **Ours (fast)** (see Section 3.5). Many compared methods include additional training data or employ models that have been pre-trained on segmentation data. For fair comparison we classify methods into two categories: “seg” for methods employing segmentation networks, pre-trained on e.g. PASCAL [14] or MS-COCO [28] and “synth” for methods that perform additional training on synthetic VOS data generated from image segmentation datasets.

**YouTube-VOS** [48]: The official YouTube-VOS validation dataset has 474 sequences with objects from 91 classes. Out of these, 26 classes are not present in the training set. We provide results for **Ours** and **Ours (fast)**, both trained on the YouTube-VOS 2018 training set. We compare our method with the results reported in [47], that were obtained by retraining the methods on YouTube-VOS. Additionally, we compare to PReMVOS [29], AGAME [22], RVOS [42] and STM [34]. The results are reported in Table 2.

Among the methods using additional training data, OSVOS [4], OnAVOS [43] and PReMVOS employ first-frame fine-tuning, leading to inferior frame-rates below 0.3 FPS. In addition to fine-tuning, PReMVOS constitutes a highly complex framework, encompassing multiple components and cues: mask-region proposals, optical flow based mask predictions, re-identification, merging and tracking modules. In contrast, our approach is simple, consisting of a single network together with a light-weight target model. Remarkably, our approach significantly outperforms PReMVOS by a relative margin of 5.2%, yielding a final $\mathcal{J}$-

<table>
<thead>
<tr>
<th>Version</th>
<th>$D$</th>
<th>$S$</th>
<th>Update</th>
<th>$\mathcal{J}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base net</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>49.8</td>
</tr>
<tr>
<td>F-T.</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>58.9</td>
</tr>
<tr>
<td>D-only - no update</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>58.3</td>
</tr>
<tr>
<td>D-only</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>59.6</td>
</tr>
<tr>
<td>Ours - no update</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>67.9</td>
</tr>
<tr>
<td>Ours</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>71.4</td>
</tr>
</tbody>
</table>

Table 1. Abblative study on a validation split of 300 sequences from the YouTube-VOS train set. We analyze the different components of our approach, where $D$ and $S$ denote the target model and segmentation network respectively. Further, “Update” indicates if the target model update is enabled. Our target model $D$ outperforms the Base net and is comparable to first-frame fine-tuning (“F-T.”) even with updates are disabled. Further, the segmentation network significantly improves the raw predictions from the target model $D$. Finally, the best performance is obtained when additionally updating target model $D$. |
score of 72.1. The recent STM method has the highest performance, employing feature matching with a dynamic memory to predict the target.

RVOS is trained only on YouTube-VOS, achieving a $J$-score of 56.8 by employing recurrent networks. In addition to recurrent networks, S2S employs first-frame fine-tuning, achieving a $J$-score of 64.4 with a significantly slower frame-rate compared to RVOS. In AGAME a generative appearance model is employed, resulting in a $J$-score of 66.1. We further report results from a version of STM (YV18), where training has been performed solely on YouTube-VOS. This significantly degrades the performance to a $J$-score of 68.2. Ours outperforms all previous methods when only video data from YouTube-VOS has been used for training. We believe that, since our target model already provides robust predictions of the target on its own, our approach can achieve high performance without extensive training on additional data. Notably, Ours-fast, maintains an impressive $J$-score of 65.7, while being significantly faster than all previous methods at 41.3 FPS.

**DAVIS 2017** [37]: The validation set for DAVIS 2017 contains 30 sequences. We provide results for Ours and Ours-fast, trained a combination of the YouTube-VOS and DAVIS 2017 train splits, such that DAVIS 2017 is traversed eight times per epoch, and YouTube-VOS once. We report the results on DAVIS 2017 in Table 3. As in the YouTube-VOS comparison above, we categorize the methods with respect to usage of training data. Since our approaches (Ours and Ours-fast) and AGAME [22], RVOS [42], STM [34] and FEELVOS [44] all include the YouTube-VOS during training, we add a third category denoted “yv”.

OnAVOS [43], OSVOS-S [30], MGCRN [19], PReMVOS [29] employ extensive fine-tuning on the first-frame, experiencing impractical segmentation speed. The methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Data</th>
<th>$\mathcal{G}$</th>
<th>$\mathcal{J}$</th>
<th>$\mathcal{F}$</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>seg</td>
<td>synth</td>
<td>overall</td>
<td>seen</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td>-</td>
<td>-</td>
<td>72.1</td>
<td>72.3</td>
</tr>
<tr>
<td>Ours-fast</td>
<td></td>
<td>-</td>
<td>-</td>
<td>72.1</td>
<td>72.3</td>
</tr>
<tr>
<td>STM (YV18) [34]</td>
<td>-</td>
<td>67.5</td>
<td>66.8</td>
<td>58.4</td>
<td>71.3</td>
</tr>
<tr>
<td>AGAME [22]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>68.2</td>
<td>-</td>
</tr>
<tr>
<td>RVOS [42]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>66.1</td>
<td>67.8</td>
</tr>
<tr>
<td>RVOS (DV17)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>56.8</td>
<td>63.6</td>
</tr>
<tr>
<td>S2S [47]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>64.4</td>
<td>71.0</td>
</tr>
<tr>
<td>ST (YV18) [34]</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>79.4</td>
<td>79.7</td>
</tr>
<tr>
<td>PReMVOS [29]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>66.9</td>
<td>71.4</td>
</tr>
<tr>
<td>OnAVOS [43]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>55.2</td>
<td>60.1</td>
</tr>
<tr>
<td>OSVOS-S [30]</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>58.8</td>
<td>59.8</td>
</tr>
<tr>
<td>Ours-fast</td>
<td></td>
<td>✓</td>
<td>-</td>
<td>79.4</td>
<td>79.7</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td>✓</td>
<td>-</td>
<td>79.4</td>
<td>79.7</td>
</tr>
<tr>
<td>RVOS [42]</td>
<td></td>
<td>-</td>
<td>✓</td>
<td>70.2</td>
<td>70.2</td>
</tr>
<tr>
<td>STM (DV17)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>51.0</td>
<td>51.0</td>
</tr>
<tr>
<td>MGCRN [19]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>51.0</td>
<td>51.0</td>
</tr>
</tbody>
</table>

Table 2. State-of-the-art comparison on the large-scale YouTube-VOS validation dataset, containing 474 videos. The results of our approach were obtained through the official evaluation server. We report the mean Jaccard ($\mathcal{J}$) and boundary ($\mathcal{F}$) scores for object classes that are seen and unseen in the training set, along with the overall mean ($\mathcal{G}$). "seg" and "synth" indicate whether pre-trained segmentation models or additional data has been used during training.

We further report results from a version of STM and second best entries are shown in red and blue respectively. In addition to Ours and Ours-fast, we report the results of our approach when trained on only DAVIS 2017, in Ours (DV17). Our approach outperforms compared methods with practical frame-rates. Furthermore, we achieve competitive results when trained with only DAVIS 2017, owing to our discriminative target model.

**Table 3. State-of-the-art comparison on DAVIS 2017 and DAVIS 2016 validation sets. The columns with “yv”, “seg”, and “synth” indicate whether YouTube-VOS, pre-trained segmentation models or additional synthetic data has been used during training. The best and second best entries are shown in red and blue respectively.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Data</th>
<th>$\mathcal{J}$</th>
<th>$\mathcal{F}$</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>yv</td>
<td>seg</td>
<td>synth</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours (DV17)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AGAME (DV17) [22]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FAVOS [7]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STM (DV17)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RVOS [42]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STM (DV17)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours-fast</td>
<td></td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RVOS [42]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STM (DV17)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RVOS [42]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STM (DV17)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours-fast</td>
<td></td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RVOS [42]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STM (DV17)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RVOS [42]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STM (DV17)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3. State-of-the-art comparison on DAVIS 2017 and DAVIS 2016 validation sets. The columns with “yv”, “seg”, and “synth” indicate whether YouTube-VOS, pre-trained segmentation models or additional synthetic data has been used during training. The best and second best entries are shown in red and blue respectively. In addition to Ours and Ours-fast, we report the results of our approach when trained on only DAVIS 2017, in Ours (DV17). Our approach outperforms compared methods with practical frame-rates. Furthermore, we achieve competitive results when trained with only DAVIS 2017, owing to our discriminative target model.

We also evaluate our approach when only trained on DAVIS 2017, denoted Ours (DV17). We compare this approach to the methods FAVOS [7], AGAME (DV17) [22] and STM (DV17) [34], which have also only been trained on the DAVIS 2017 train split. Our method significantly outperforms all these methods with a $\mathcal{J}$&$\mathcal{F}$ score of 76.7. In addition, Ours-fast is significantly faster than all previous approaches, maintaining a $\mathcal{J}$&$\mathcal{F}$ score of 70.2. Our method is only outperformed by PReMVOS and the recent STM, achieving $\mathcal{J}$&$\mathcal{F}$ scores of 77.8 and 81.8 respectively. PReMVOS, however, suffer from extremely slow frame rates: approximately 500 times lower than ours. Moreover, our approach outperforms PReMVOS on the more challenging and large scale YouTube-VOS (Table 2) by a large margin.
Figure 4. Examples of three sequences from DAVIS 2017, demonstrating how our method performs under significant appearance changes compared to ground truth, OSVOS-S [30], AGAME [22], FEELVOS [44] and RGMP [33].

Figure 5. Qualitative results of our method, showing both target model score maps and the output segmentation masks. The top three rows are success cases and the last two represents failures when objects are too thin or appear too similar.

4.3. Qualitative Analysis

State-of-the-art: We compare our approach to some state-of-the-art methods in Figure 4. By using an early and a late frame from each video sequence, we study how the methods cope with large target deformations over time. In the first sequence (first and second row), the fine-tuning based OSVOS-S struggles as the target pose changes. While the mask-propagation in RGMP and generative appearance model in AGAME are accurate on fine details, they both fail to segment the red target, possibly due to occlusions. In the second and third sequences (rows three to six), all of the above methods fail to robustly segment the different targets. In contrast, our method accurately segments all targets in these challenging video sequences.

Target model: Some examples of the coarse segmentation scores and the final segmentation output are visualized in Figures 1 and 5. In most cases, the target model provides robust segmentation scores of the target object. It however struggles in some cases where the target object contains thin or small structures or details. An example is the challenging kite lines in the kite-surfing sequence, which are not accurately segmented. This is likely due to the coarse feature maps the target model is operating on. It also have problems separating almost identical targets such as the sheep. On the other hand, the model successfully handles very similar targets as in the gold-fish sequence (row 2 in Figure 1).

5. Conclusion

We propose video object segmentation approach, integrating a light-weight but highly discriminative target appearance model and a segmentation network. We find that despite its simplicity, a linear discriminative model is capable of generating robust target predictions. The segmentation network converts the predictions into high-quality object segmentations. The target model is efficiently trained during inference. Our method operates at high frame-rates and achieves state-of-the-art performance on the YouTube-VOS dataset and competitive results on DAVIS 2017 despite trained on limited data.

Acknowledgments: This work was supported by the ELLIIT Excellence Center, the Wallenberg AI, Autonomous Systems and Software Program (WASP) and the SSF project Symbolcloud.
References


[26] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Flugfelder, Luká Cehovin Zajc, Tomá Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, Gus-


