Associating Objects with Transformers for Video Object Segmentation

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper investigates how to realize better and more efficient embedding learning to tackle the semi-supervised video object segmentation under challenging multi-object scenarios. The state-of-the-art methods learn to decode features with a single positive object and thus have to match and segment each target separately under multi-object scenarios, consuming multiple times computing resources. To solve the problem, we propose an Associating Objects with Transformers (AOT) approach to match and decode multiple objects uniformly. In detail, AOT employs an identification mechanism to associate multiple targets into the same high-dimensional embedding space. Thus, we can simultaneously process the matching and segmentation decoding of multiple objects as efficiently as processing a single object. For sufficiently modeling multi-object association, a Long Short-Term Transformer is designed for constructing hierarchical matching and propagation. We conduct extensive experiments on both multi-object and single-object benchmarks to examine AOT variant networks with different complexities. Particularly, our AOT-L outperforms all the state-of-the-art competitors on three popular benchmarks, *i.e.*, YouTube-VOS (83.7% $\mathcal{J}\&\mathcal{F}$), DAVIS 2017 (83.0%), and DAVIS 2016 (91.0%), while keeping better multi-object efficiency. The code will be publicly available.

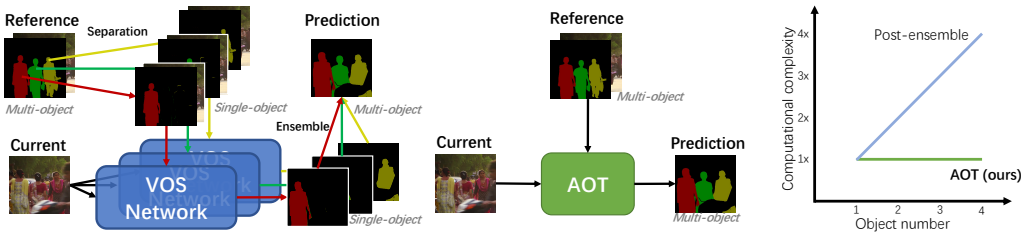
1 Introduction

Video Object Segmentation (VOS) is a fundamental task in video understanding with many potential applications, including augmented reality [1] and self-driving cars [2]. The goal of semi-supervised VOS, the main task in this paper, is to track and segment object(s) across an entire video sequence based on the object mask(s) given at the first frame.

Thanks to the recent advance of deep neural networks, many deep learning based VOS algorithms have been proposed recently and achieved promising performance. STM [3] and its following works [4, 5] leverage a memory network to store and read the target features of predicted past frames and apply a non-local attention mechanism to match the target in the current frame. FEELVOS [6] and CFBI [7] utilize global and local matching mechanisms to match target pixels or patches from both the first and the previous frames to the current frame.

Even though the above methods have achieved significant progress, the above methods learn to decode scene features that contain a single positive object. Thus under a multi-object scenario, they have to match each object independently and ensemble all the single-object predictions into a multi-object segmentation, as shown in Fig. 1a. Such a post-ensemble manner eases network architectures' design since the networks are not required to adapt the parameters or structures for different object numbers. However, modeling multiple objects independently, instead of uniformly, is inefficient in exploring multi-object contextual information to learn a more robust feature representation for VOS. In addition,

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) Post-ensemble (b) Associating objects (ours) (c) Comparison

Figure 1: The state-of-the-art VOS methods (*e.g.*, [4, 7]) processes multi-object scenarios in a post-ensemble manner (a). In contrast, our AOT associates and decodes multiple objects uniformly (b), leading to better efficiency (c).

processing multiple objects separately yet in parallel requires multiple times the amount of GPU memory and computation for processing a single object. This problem restricts the training and application of VOS under multi-object scenarios, especially when computing resources are limited.

To solve the problem, Fig. 1b demonstrates a feasible approach to associate and decode multiple objects uniformly in an end-to-end framework. Hence, we propose an Associating Objects with Transformers (AOT) approach to match and decode multiple targets uniformly. First, an identification mechanism is proposed to assign each target a unique identity and embed multiple targets into the same feature space. Hence, the network can learn the association or correlation among all the targets. Moreover, the multi-object segmentation can be directly decoded by utilizing assigned identity information. Second, a Long Short-Term Transformer (LSTT) is designed for constructing hierarchical object matching and propagation. Each LSTT block utilizes a long-term attention for matching with the first frame's embedding and a short-term attention for matching with several nearby frames' embeddings. Compared to the methods [3, 4] utilizing only one attention layer, we found hierarchical attention structures are more effective in associating multiple objects.

We conduct extensive experiments on two popular multi-object benchmarks for VOS, *i.e.*, YouTube-VOS [8] and DAVIS 2017 [9], to validate the effectiveness and efficiency of the proposed AOT. Even using the light-weight Mobilenet-V2 [10] as the backbone encoder, the AOT variant networks achieve superior performance on the validation 2018 & 2019 splits of the large-scale YouTube-VOS (ours, $\mathcal{J}\&\mathcal{F}$ **82.6~83.7**% & **82.2~83.6**%) while keeping faster multi-object run-time (**12.5~6.3**FPS) compared to the state-of-the-art competitors (*e.g.*, CFBI [7], 81.4% & 81.0%, 3.4FPS). We also achieve new state-of-the-art performance on both the DAVIS-2017 validation (**83.0**%) and testing (**78.8**%) splits. Moreover, AOT is effective under single-object scenarios as well and outperforms previous state-of-the-art methods on DAVIS 2016 [11] (**91.0**%), a popular single-object benchmark.

Overall, our contributions are summarized as follows:

- We propose an identification mechanism to associate and decode multiple targets uniformly for VOS. For the first time, multi-object training and inference can be efficient as single-object ones, as demonstrated in Fig. 1c.
- Based on the identification mechanism, we design a new efficient VOS framework, *i.e.*, Long Short-Term Transformer (LSTT), for constructing hierarchical multi-object matching and propagation. LSTT achieves superior performance on VOS benchmarks [8, 9, 11] while maintaining better efficiency than previous state-of-the-art methods. To the best of our knowledge, LSTT is the first hierarchical attention-based framework to apply transformers [12] to VOS.

2 Related Work

Semi-supervised Video Object Segmentation. Given one or more annotated frames (the first frame in general), semi-supervised VOS methods propagate the manual labeling to the entire video sequence. Traditional methods often solve an optimization problem with an energy defined over a graph structure [13, 14, 15]. Methods in recent years are always based on deep convolutional neural networks, leading to more robust and accurate results.

Some methods rely on fine-tuning the networks at test time to make convolutional segmentation networks focus on a specific target object. Among them, OSVOS [16] and MoNet [17] fine-tune pre-trained networks on the first-frame ground-truth at test time. OnAVOS [18] extends the first-frame fine-tuning by introducing an online adaptation mechanism, *i.e.*, fine-tuning networks with highly-confident predictions during inference online. Following these approaches, MaskTrack [19] and PReM [20] utilize optical flow to help propagate the segmentation mask from one frame to the next. Despite achieving promising results, the test-time fine-tuning restricts the network efficiency.

Recent works aim to achieve a better run-time and avoid using online fine-tuning. OSMN [21] employs one convolutional network to extract object embedding and another one to guide segmentation predictions. PML [22] learns pixel-wise embedding with a nearest neighbor classifier. Similar to PML, VideoMatch [23] uses a soft matching layer that maps the pixels of the current frame to the first frame in a learned embedding space. Following PML and VideoMatch, FEELVOS [6] and CFBI [7] extend the pixel-level matching mechanism by additionally matching between the current frame and the previous frame. RGMP [24] also gathers guidance information from both the first frame and the previous frame but uses a siamese encoder with two shared streams. STM [3] and its following works (*e.g.*, EGMN [5] and KMN [4]) leverage a memory network to embed past-frame predictions into memory and apply a non-local attention mechanism on the memory to decode the segmentation of the current frame. Instead of using attention mechanisms, LWL [25] proposes to use an online few-shot learner but generalizes worse than the above methods on the small-scale DAVIS [9].

The above methods learn to decode features with a single positive object and thus have to match and segment each target separately under multi-object scenarios, consuming multiple times computing resources of single-object cases. The problem restricts the application and development of the VOS with multiple targets. Hence, we propose our AOT to associate and decode multiple targets uniformly and simultaneously, as efficiently as processing a single object.

Visual Transformers. Transformers [12] was proposed to build hierarchical attention-based networks for machine translation. Similar to Non-local Neural Networks [26], transformer blocks compute correlation with all the input elements and aggregate their information by using attention mechanisms [27]. Compared to RNNs, transformer networks model global correlation or attention in parallel, leading to better memory efficiency, and thus have been widely used in natural language processing (NLP) tasks [28, 29, 30]. Recently, transformer blocks were introduced to many computer vision tasks, such as image classification [31], object detection [32], and image generation [33], and have shown promising performance compared to CNN-based networks.

Many VOS methods [3, 4, 5] have utilized attention mechanisms to match the object features and propagate the segmentation mask from past frames to the current frames. Nevertheless, these methods consider only one positive target in the attention processes, and how to build hierarchical attention-based propagation has been rarely studied. In this paper, we carefully design a long short-term transformer block, which can effectively construct multi-object matching and propagation within hierarchical structures for VOS.

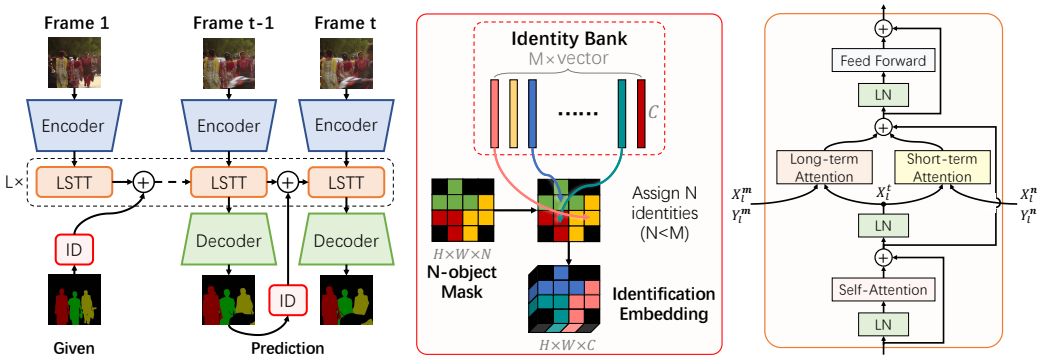
3 Revisit Previous Solutions for Video Object Segmentation

In VOS, many common video scenarios have multiple targets or objects required for tracking and segmenting. Benefit from deep networks, current state-of-the-art VOS methods [3, 7] have achieved promising performance. Nevertheless, these methods focus on matching and decoding a single object. Under a multi-object scenario, they thus have to match each object independently and ensemble all the single-object predictions into a multi-object prediction, as demonstrated in Fig. 1a. Let $F^{\mathcal{N}}$ denotes a VOS network for predicting single-object segmentation, and A is an ensemble function such as $softmax$ or the soft aggregation [3], the formula of such a post-ensemble manner for processing N objects is like,

$$Y' = A(F^{\mathcal{N}}(I^t, I^{\mathbf{m}}, Y_1^{\mathbf{m}}), ..., F^{\mathcal{N}}(I^t, I^{\mathbf{m}}, Y_N^{\mathbf{m}})), \tag{1}$$

where I^t and $I^{\mathbf{m}}$ denote the image of the current frame and memory frames respectively, and $\{Y_1^{\mathbf{m}}, ..., Y_N^{\mathbf{m}}\}$ are the memory masks (containing the given reference mask and past predicted masks) of all the N objects. This manner extends networks designed for single-object VOS into multi-object applications, so there is no need to adapt the network for different object numbers.

Although the above post-ensemble manner is prevalent and straightforward in the VOS field, processing multiple objects separately yet in parallel requires multiple times the amount of GPU memory and



(a) Overview (b) Identity assignment (c) l-th LSTT block

Figure 2: (a) The overview of our Associating Objects with Transformers (AOT). The multi-object masks are embedded by using our Identification mechanism. Moreover, a L-layer Long Short-Term Transformer is responsible for matching multiple objects uniformly and hierarchically. (b) An illustration of the IDentity assignment (ID) designed for transferring a N-object mask into an identification embedding. (c) The structure of an LSTT block. LN: layer normalization [34].

computation for matching a single object and decoding the segmentation. This problem restricts the training and application of VOS under multi-object scenarios when computing resources are limited. To make the multi-object training and inference as efficient as single-object ones, an expected solution should be capable of associating and decoding multiple objects uniformly instead of individually. To achieve such an objective, we propose an identification mechanism to embed the masks of any number (required to be smaller than a pre-defined large number) of targets into the same high-dimensional space. Based on the identification mechanism, a novel and efficient framework, *i.e.*, Associating Objects with Transformers (AOT), is designed for propagating all the object embeddings uniformly and hierarchically, from memory frames to the current frame.

As shown in Fig. 1b, our AOT associates and segments multiple objects within an end-to-end framework. For the first time, processing multiple objects can be as efficient as processing a single object (Fig. 1c). Compared to previous methods, our training under multi-object scenarios is also more efficient since AOT can associate multiple object regions and learn contrastive feature embeddings among them uniformly.

4 Associating Objects with Transformers

In this section, we introduce our identification mechanism proposed for efficient multi-object VOS. Then, we design a new VOS framework, *i.e.*, long short-term transformer, based on the identification mechanism for constructing hierarchical multi-object matching and propagation.

4.1 Identification Mechanism for Multi-object Association

Many recent VOS methods [3, 4, 5] utilized attention mechanisms and achieved promising results. To formulate, we define $Q \in \mathbb{R}^{HW \times C}$, $K \in \mathbb{R}^{THW \times C}$, and $V \in \mathbb{R}^{THW \times C}$ as the query embedding of the current frame, the key embedding of the memory frames, and the value embedding of the memory frames respectively, where T, H, W, C denote the temporal, height, width, and channel dimensions. The formula of a common attention-based matching and propagation is,

$$Att(Q, K, V) = Corr(Q, K)V = softmax(\frac{QK^{tr}}{\sqrt{C}})V, \tag{2}$$

where a matching map is calculated by the correlation function $Corr$, and then the value embedding, V, will be propagated into each location of the current frame.

In the common single-object propagation [3], the binary mask information in memory frames is embedded into V with an additional memory encoder network and thus can also be propagated to the

current frame by using Eq. 2. A convolutional decoder network following the propagated feature will decode the aggregated feature and predict the single-object probability logit of the current frame.

The main problem of propagating and decoding multi-object mask information in an end-to-end network is how to adapt the network to different target numbers. To overcome this problem, we propose an identification mechanism consisting of identification embedding and decoding based on attention mechanisms.

First, an **Identification Embedding** mechanism is proposed to embed the masks of multiple different targets into the same feature space for propagation. As seen in Fig. 2b, we initialize an identity bank, $D \in \mathbb{R}^{M \times C}$, where M identification vectors with C dimensions are stored. For embedding multiple different target masks, each target will be randomly assigned a different identification vector. Assuming N ($N < M$) targets are in the video scenery, the formula of embedding the targets' one-hot mask, $Y \in \{0,1\}^{THW \times N}$, into a identification embedding, $E \in \mathbb{R}^{THW \times C}$, by randomly assigning identification vector from the bank D is,

$$E = ID(Y, D) = YPD, \tag{3}$$

where $P \in \{0,1\}^{M \times N}$ is a random permutation matrix, satisfying that PP^{tr} is equal to a $M \times M$ unit matrix, for randomly selecting N identification embeddings. After the ID assignment, different target has different identification embedding, and thus we can propagate all the target identification information from memory frames to the current frame by attaching the identification embedding E with the attention value V, *i.e.*,

$$V' = AttID(Q, K, V, Y|D) = Att(Q, K, V + ID(Y, D)) = Att(Q, K, V + E), \tag{4}$$

where $V' \in \mathbb{R}^{HW \times C}$ aggregates all the multiple targets' embeddings from the propagation.

For **Identification Decoding**, *i.e.*, predicting all the targets' probabilities from the aggregated feature V', we firstly predict the probability logit for every identity in the bank D by employing a convolutional decoding network $F^{\mathcal{D}}$, and then select the assigned ones and calculate the probabilities, *i.e.*,

$$Y' = softmax(PF^{\mathcal{D}}(V')) = softmax(PL^{D}), \tag{5}$$

where $L^D \in \mathbb{R}^{HW \times M}$ is all the M identities' probability logits, P is the same as the selecting matrix used in the identity assignment (Eq. 3), and $Y' \in [0,1]^{HW \times N}$ is the probability prediction of all the N targets.

For training, common multi-class segmentation losses, such as cross-entropy loss, can be used to optimize the multi-object Y' regarding the ground-truth labels. The identity bank D is trainable and randomly initialized at the training beginning. To ensure that all the identification vectors have the same opportunity to compete with each other, we randomly reinitialize the identification selecting matrix P in each video sample and each optimization iteration.

4.2 Long Short-Term Transformer for Hierarchical Matching and Propagation

Previous methods [3, 4] always utilize only one layer of attention (Eq. 2) to aggregate single-object information. In our identification-based multi-object pipeline, we found that a single attention layer cannot fully model multi-object association, which naturally should be more complicated than single-object processes. Thus, we consider constructing hierarchical matching and propagation by using a series of attention layers. Recently, transformer blocks [12] have been demonstrated to be stable and promising in constructing hierarchical attention structures in visual tasks [31, 32]. Based on transformer blocks, we carefully design a Long Short-Term Transformer (LSTT) block for multi-object VOS.

Following the common transformer blocks [12, 28], LSTT firstly employs a self-attention layer, which is responsible for learning the association or correlation among the targets within the current frame. Then, LSTT additionally introduces a long-term attention, for aggregating targets' information from long-term memory frames and a short-term attention, for learning temporal smoothness from nearby short-term frames. The final module is a common 2-layer feed-forward MLP with GELU [35] non-linearity in between. Fig. 2c shows the structure of an LSTT block. Notably, all these attention modules are implemented in the form of the multi-head attention [12], *i.e.*, multiple attention modules followed by concatenation and a linear projection. Nevertheless, we only introduce their single-head formulas below for the sake of simplicity.

Long-Term Attention is responsible for aggregating targets' information from past memory frames, which contains the reference frame and stored predicted frames, to the current frame. Since the temporal intervals between the current frame and past frames are variable and can be long-term, the temporal smoothness is difficult to guaranteed. Thus, the long-term attention employs a non-local attention like Eq. 2. Let $X_l^t \in \mathbb{R}^{HW \times C}$ denotes the input feature embedding at time t and in block l, where $l \in \{1, ..., L\}$ is the block index of LSTT, the formula of the long-term attention is,

$$AttLT(X_l^t, X_l^{\mathbf{m}}, Y^{\mathbf{m}}) = AttID(X_l^t W_l^K, X_l^{\mathbf{m}} W_l^K, X_l^{\mathbf{m}} W_l^V, Y^{\mathbf{m}} | D), \tag{6}$$

where $X_l^{\mathbf{m}} = Concat(X_l^{m_1}, ..., X_l^{m_T})$ and $Y^{\mathbf{m}} = Concat(Y^{m_1}, ..., Y^{m_T})$ are the input feature embeddings and target masks of memory frames with indices $\mathbf{m} = \{m_1, ..., m_T\}$. Besides, $W_l^K \in \mathbb{R}^{C \times C_k}$ and $W_l^V \in \mathbb{R}^{C \times C_v}$ are trainable parameters of the space projections for matching and propagation, respectively. Instead of using different projections for X_l^t and $X_l^{\mathbf{m}}$, we found the training of LSTT is more stable with a siamese-like matching, *i.e.*, matching between the features within the same embedding space (l-th features with the same projection of W_l^K).

Short-Term Attention is employed for aggregating information in a spatial-temporal neighbourhood for each current-frame location. Intuitively, the image changes across several contiguous video frames are always smooth and continuous. Thus, the target matching and propagation in contiguous frames can be restricted in a small spatial-temporal neighborhood, leading to better efficiency than non-local processes. Considering n neighbouring frames with indices $\mathbf{n} = \{t-1, ..., t-n\}$ are in the spatial-temporal neighbourhood, the features and masks of these frames are $X_l^{\mathbf{n}} = Concat(X_l^{t-1}, ..., X_l^{t-n})$ and $Y^{\mathbf{n}} = Concat(Y^{t-1}, ..., Y^{t-n})$, and then the formula of the short-term attention at each spatial location p is,

$$AttST(X_l^t, X_l^{\mathbf{n}}, Y^{\mathbf{n}} | p) = AttLT(X_{l,p}^t, X_{l,\mathcal{N}(p)}^{\mathbf{n}}, Y_{l,\mathcal{N}(p)}^{\mathbf{n}}), \tag{7}$$

where $X_{l,p}^t \in \mathbb{R}^{1 \times C}$ is the feature of X_l^t at location p, $\mathcal{N}(p)$ is a $\lambda \times \lambda$ spatial neighbourhood centered at location p, and thus $X_{l,\mathcal{N}(p)}^{\mathbf{n}}$ and $Y_{l,\mathcal{N}(p)}^{\mathbf{n}}$ are the features and masks of the spatial-temporal neighbourhood, respectively, with a shape of $n\lambda^2 \times C$ or $n\lambda^2 \times N$.

When extracting features of the first frame $t = 1$, there is no memory frames or previous frames, and hence we use X_l^1 to replace $X_l^{\mathbf{m}}$ and $X_l^{\mathbf{n}}$. In other words, the long-term attention and the short-term attention are changed into self-attentions without adjusting the network structures and parameters.

5 Implementation Details

Network Details: For sufficiently validating the effectiveness of our identification mechanism and LSTT, we use only light-weight backbone encoder, MobileNet-V2 [10], and decoder, FPN [36] with Group Normalization [37]. Following [32], we increase the final resolution of the encoder to $1/16$ by adding a dilation to the last stage of the encoder and removing a stride from the first convolution of this stage. The features are flattened into sequences before LSTT. In LSTT, the basic channel dimension is 256, and the head number is set to 8 for all the attention modules. In our default setting of the short-term memory $\mathbf{n}$, only the previous $(t-1)$ frame is considered, which is similar to the local matching strategy [6, 7], and the spatial neighborhood size λ is set to 7. After LSTT, the features are reshaped into 2D shapes. Then, the FPN decoder progressively increases the feature resolution from $1/16$ to $1/4$ and decreases the channel dimension from 256 to 64 before one final output layer and our identification decoding. The number of identification vectors, M, is set to 10, which is consistent with the maximum object number in the benchmarks [8, 9].

Architecture Variants: We build several AOT variant networks with different LSTT layer number L or long-term memory size $\mathbf{m}$. The hyper-parameters of these variants are: **(1) AOT-Small**: $L = 2$, $\mathbf{m} = \{1\}$; **(2) AOT-Base**: $L = 3$, $\mathbf{m} = \{1\}$; **(3) AOT-Large**: $L = 3$, $\mathbf{m} = \{1, 6, 11, 16, ...\}$.

AOT-S is a small model with only 2 layers of LSTT block. AOT-B and AOT-L use more LSTT layers, *i.e.*, 3 layers. In AOT-S and AOT-B, only the first frame is considered into long-term memory, which is similar to [6, 7], leading to a smooth efficiency. In AOT-L, the predicted frames are stored into long-term memory per 5 frames, following the memory reading strategy [3, 4, 5].

Training Details: Following [3, 4, 5, 24], the training stage is divided into two phases: (1) pre-training on sythetic video sequence generated from static image datasets [38, 39, 40, 41, 42] by randomly applying multiple image augmentations [24]. (2) main training on the VOS benchmarks [8, 9] by randomly applying video augmentations [7].

Table 1: The quantitative evaluation on multi-object benchmarks, YouTube-VOS [8] and DAVIS 2017 [9]. **Y**: additionally using YouTube-VOS for training. *: using 600p resolution instead of 480p during inference. ‡: timing extrapolated from single-object speed assuming linear scaling in the number of objects.

(a) YouTube-VOS

		Seen		Unseen		
Methods	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	FPS
Validation 2018 Split						
AG[CVPR19] [45]	66.1	67.8	-	60.8	-	-
PReM[ACCV18] [20]	66.9	71.4	75.9	56.5	63.7	0.17
BoLT[arXiv19] [46]	71.1	71.6	-	64.3	-	0.74
STM[ICCV19] [3]	79.4	79.7	84.2	72.8	80.9	-
EGMN[ECCV20] [5]	80.2	80.7	85.1	74.0	80.9	-
KMN[ECCV20] [4]	81.4	81.4	85.6	75.3	83.3	-
CFBI[ECCV20] [7]	81.4	81.1	85.8	75.3	83.4	3.4
LWL[ECCV20] [25]	81.5	80.4	84.9	76.4	84.4	-
AOT-S	82.6	82.0	86.7	76.6	85.0	**12.5**
AOT-B	83.2	**82.6**	87.4	77.3	85.6	8.0
AOT-L	**83.7**	82.5	**87.5**	**77.9**	**86.7**	6.3
Validation 2019 Split						
CFBI[ECCV20] [7]	81.0	80.6	85.1	75.2	83.0	3.4
AOT-S	82.2	81.3	85.9	76.6	84.9	**12.5**
AOT-B	83.3	**82.5**	**87.0**	77.8	86.0	8.0
AOT-L	**83.6**	82.2	86.9	**78.3**	**86.9**	6.3

(b) DAVIS 2017

Methods	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	FPS
Validation 2017 Split				
STM [3] (**Y**)	81.8	79.2	84.3	3.1‡
CFBI [7] (**Y**)	81.9	79.3	84.5	5.9
KMN [4]	76.0	74.2	77.8	4.2‡
KMN [4] (**Y**)	82.8	80.0	85.6	4.2‡
AOT-S	79.2	76.4	82.0	**18.7**
AOT-S (**Y**)	81.0	78.5	83.4	**18.7**
AOT-B (**Y**)	82.1	79.4	84.8	12.3
AOT-L (**Y**)	**83.0**	**80.3**	**85.7**	8.0
Testing 2017 Split				
STM* [3] (**Y**)	72.2	69.3	75.2	-
CFBI [7] (**Y**)	75.0	71.4	78.7	5.3
CFBI* [7] (**Y**)	76.6	73.0	80.1	2.9
KMN* [4] (**Y**)	77.2	74.1	80.3	-
AOT-S (**Y**)	73.6	69.7	77.4	**18.7**
AOT-B (**Y**)	75.5	71.8	79.1	12.3
AOT-L (**Y**)	78.4	74.8	82.1	8.0
AOT-L* (**Y**)	**78.8**	**75.3**	**82.3**	4.1

All the videos are firstly down-sampled to 480p resolution, and the cropped window size is 465$\times$ 465. For optimization, we adopt the AdamW [43] optimizer and apply a bootstrapped cross-entropy loss with the sequential training strategy [7], whose sequence length is set to 5. The statistics of BN [44] in the encoder are also frozen for stabilizing the training.

The batch size is 16 and distributed on 4 Tesla V100 GPUs. For pre-training, we use an initial learning rate of 4×10^{-4} for 100,000 steps. For main training, the initial learning rate is set to 2×10^{-4}, and the training steps are 100,000 for YouTube-VOS or 50,000 for DAVIS [7]. The learning rate gradually decays to 10^{-5} in a polynomial manner [7]. More details are placed in the supplementary material.

6 Experimental Results

We evaluate our method on popular multi-object benchmarks, YouTube-VOS [8] and DAVIS 2017 [9], and single-object benchmark, DAVIS 2016 [11]. For YouTube-VOS experiments, we train our models on the YouTube-VOS 2019 training split. For DAVIS, we train on the DAVIS-2017 training split. When evaluating YouTube-VOS, all the videos are restricted to be no more than $1.3 \times 480p$ resolution, consistent with the training stage. As to DAVIS, the default 480p resolution is used.

The evaluation metric is the $\mathcal{J}$ score, calculated as the average Intersect over Union (IoU) score between the prediction and the ground truth mask, and the $\mathcal{F}$ score, calculated as an average boundary similarity measure between the boundary of the prediction and the ground truth, and their mean value, denoted as $\mathcal{J}\&\mathcal{F}$. We evaluate all the results on official evaluation servers or with official tools.

6.1 Compare with the State-of-the-art Methods

YouTube-VOS [8] is the latest large-scale benchmark for multi-object video segmentation and is about 37 times larger than DAVIS 2017 (120 videos). Specifically, YouTube-VOS contains 3471 videos in the training split with 65 categories and 474/507 videos in the validation 2018/2019 split with additional 26 unseen categories. The unseen categories do not exist in the training split in order to evaluate the generalization ability of algorithms.

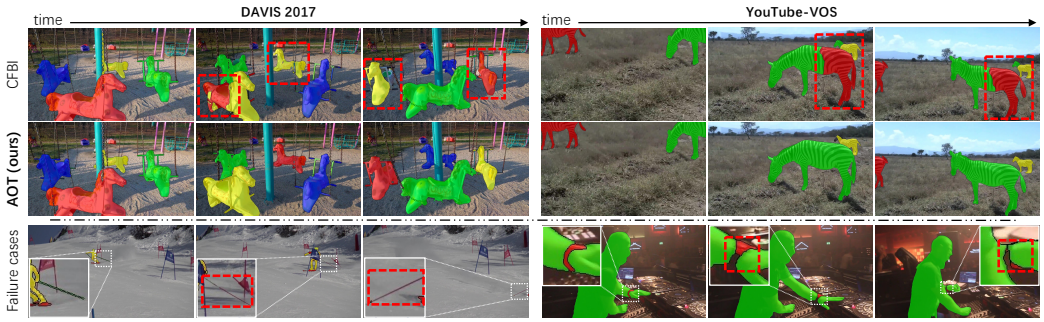


Figure 3: Qualitative results. (top) Compared with CFBI [7], AOT performs better when segmenting multiple highly similar objects (*carousels* and *zebras*). (bottom) AOT fails to segment some tiny objects (*ski poles* and *watch*) since AOT has no specific design for processing rare tiny objects.

As shown in Table 1a, AOT variants achieve superior performance on YouTube-VOS compared to the previous state-of-the-art methods. With our identification mechanism, AOT-S (82.6% $\mathcal{J}\&\mathcal{F}$) surpasses CFBI [7] (81.4%) by +1.2% while running about $4\times$ faster (12.5 *vs* 3.4FPS). By using more LSTT blocks, AOT-B effectively improves the performance to 83.2%. Moreover, by utilizing the memory reading strategy, the unseen scores of AOT can be further improved, and our AOT-L (**83.7%/83.6%**, 6.3FPS) significantly outperforms the previous methods (*e.g.*, CFBI, 81.4%/81.0%, 3.4FPS) on the validation 2018/2019 split while maintains an efficient speed.

DAVIS 2017 [9] is a multi-object extension of DAVIS 2016. The validation split of DAVIS 2017 consists of 30 videos with 59 objects, and the training split contains 60 videos with 138 objects. Moreover, the testing split contains 30 more challenging videos with 89 objects in total.

As shown in Table 1b, our AOT-L (**Y**) outperforms all the competitors on both the DAVIS-2017 validation (**83.0%**) and testing (**78.4%**) splits and maintains an efficient speed (8FPS). Notably, such a multi-object speed is the same as our single-object speed on DAVIS 2016. For the first time, processing multiple objects is as efficient as processing a single object using the AOT approach. We also evaluate our method without training with YouTube-VOS, and AOT-S (79.2%) performs much better than KMN [4] (76.0%) by +3.2%.

DAVIS 2016 [11] is a single-object benchmark containing 20 videos in the validation split. Although our AOT aims at improving multi-object video segmentation, we also achieve a new state-of-the-art performance on DAVIS 2016 (AOT-L (**Y**), **91.0%**). Under single-object scenarios, the multi-object superiority of AOT is limited, but AOT-L still maintains a comparable efficiency with KMN (8.0 *vs* 8.3FPS). Furthermore, our smaller variant, AOT-S (89.3%), achieves comparable performance with CFBI (89.4%) while running about $3\times$ faster (18.7 *vs* 6.3FPS).

Table 2: The quantitative evaluation on the single-object DAVIS 2016 [11].

Methods	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	FPS
STM [3] (**Y**)	89.3	88.7	89.9	6.3
CFBI [7] (**Y**)	89.4	88.3	90.5	6.3
KMN [4] (**Y**)	90.5	89.5	91.5	8.3
AOT-S (**Y**)	89.3	88.6	89.9	**18.7**
AOT-B (**Y**)	89.9	88.8	90.9	12.3
AOT-L (**Y**)	**91.0**	**89.7**	**92.3**	8.0

Qualitative results: Fig. 3 visualizes some qualitative results in comparison with CFBI [7], which only associates each object with its relative background. As demonstrated, CFBI is easier to confuse multiple highly similar objects. In contrast, our AOT tracks and segments all the targets accurately by associating all the objects uniformly. However, AOT fails to segment some tiny objects (*ski poles* and *watch*) since we do not make special designs for tiny objects. We supply more qualitative results in the supplementary material.

6.2 Ablation Study

In this section, we analyze the main components and hyper-parameters of AOT and evaluate their impact on the VOS performance in Table 3.

Identity number: The number of the identification vectors, M, have to be larger than the object number in videos. Thus, we set M to 10 in default to be consistent with the maximum object number in the benchmarks [8, 9]. As seen in Table 3a, M larger than 10 leads to worse performance since

Table 3: Ablation study. The experiments are based on AOT-S and conducted on the validation 2018 split of YouTube-VOS [8] without pre-training on synthetic videos. Self: the position embedding type used in the self-attention. Rel: use relative positional embedding [47] on the local attention.

(a) Identity number

M	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}^{seen}$	$\mathcal{J}^{unseen}$
10	**80.3**	**80.6**	**73.7**
15	79.0	79.4	72.1
20	78.3	79.4	70.8
30	77.2	78.5	70.2

(b) Local window size

λ	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}^{seen}$	$\mathcal{J}^{unseen}$
7	**80.3**	**80.6**	**73.7**
5	78.8	79.5	71.9
3	78.3	79.3	70.9
0	74.3	74.9	67.6

(c) Local frame number

n	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}^{seen}$	$\mathcal{J}^{unseen}$
1	**80.3**	**80.6**	**73.7**
2	80.0	79.8	73.7
3	79.1	80.0	72.2
0	74.3	74.9	67.6

(d) LSTT block number

L	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}^{seen}$	$\mathcal{J}^{unseen}$	FPS
2	80.3	80.6	73.7	12.5
3	**80.9**	**81.1**	**74.0**	8.0
1	77.9	78.8	71.0	**25.3**

(e) Positional embedding

Self	Rel	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}^{seen}$	$\mathcal{J}^{unseen}$
sine	✓	**80.3**	**80.6**	**73.7**
none	✓	80.1	80.4	73.5
sine	-	79.7	80.1	72.9

that (1) no training video contains so many objects; (2) embedding more than 10 objects into the space with only 256 dimensions is difficult.

Local window size: Table 3b shows that larger local window size, λ, results in better performance. Without the local attention, $\lambda = 0$, the performance of AOT significantly drops from 80.3% to 74.3%, which demonstrates the necessity of the local attention.

Local frame number: In Table 3c, we also try to employ more previous frames in the local attention, but using only the $t-1$ frame (80.3%) performs better than using 2 frames (80.0%) or 3 frames (79.1%). A possible reason is that the longer the temporal interval, the more intense the motion between frames, so it is easier to introduce more errors in the local matching when using an earlier previous frame.

LSTT block number: As shown in Table 3d, the AOT performance increases by using more LSTT blocks. Notably, the AOT with only one LSTT block (77.9%) reaches a real-time speed (25.3FPS) on YouTube-VOS, although the performance is -2.4% worse than AOT-S (80.3%). By adjusting the LSTT block number, we can flexibly balance the accuracy and speed of AOT.

Position embedding: In our default setting, we apply fixed sine spatial positional embedding to the self-attention following [32], and our local attention is equipped with learned relative positional embedding [47]. The ablation study is shown in Table 3e, where removing the sine embedding decreases the performance to 80.1% slightly. In contrast, the relative embedding is more important than the sine embedding. Without the relative embedding, the performance drops to 79.7%, which means the motion relationship between adjacent frames is helpful for the local attention. We also tried to apply learned positional embedding to the self-attention, but no positive effect was observed.

7 Conclusion

This paper proposes a novel and efficient approach for video object segmentation by associating objects with transformers and achieves superior performance on three popular benchmarks. A simple yet effective identification mechanism is proposed to associate, match, and decode all the objects uniformly under multi-object scenarios. For the first time, processing multiple objects in VOS can be efficient as processing a single object by using the identification mechanism. In addition, a long short-term transformer is designed for constructing hierarchical object matching and propagation for VOS. The hierarchical structure allows us to flexibly balance AOT between real-time speed and state-of-the-art performance by adjusting the LSTT number. We hope the identification mechanism will help ease the future study of multi-object VOS, and AOT will serve as a solid baseline.

References

[1] Ngan, K. N., H. Li. *Video segmentation and its applications*. Springer Science & Business Media, 2011.

[2] Zhang, Z., S. Fidler, R. Urtasun. Instance-level segmentation for autonomous driving with deep densely connected mrfs. In *CVPR*, pages 669–677. 2016.

[3] Oh, S. W., J.-Y. Lee, N. Xu, et al. Video object segmentation using space-time memory networks. In *ICCV*. 2019.

[4] Seong, H., J. Hyun, E. Kim. Kernelized memory network for video object segmentation. In *ECCV*. 2020.

[5] Lu, X., W. Wang, M. Danelljan, et al. Video object segmentation with episodic graph memory networks. In *ECCV*. 2020.

[6] Voigtlaender, P., Y. Chai, F. Schroff, et al. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *CVPR*, pages 9481–9490. 2019.

[7] Yang, Z., Y. Wei, Y. Yang. Collaborative video object segmentation by foreground-background integration. In *ECCV*. 2020.

[8] Xu, N., L. Yang, Y. Fan, et al. Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*, 2018.

[9] Pont-Tuset, J., F. Perazzi, S. Caelles, et al. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

[10] Sandler, M., A. Howard, M. Zhu, et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520. 2018.

[11] Perazzi, F., J. Pont-Tuset, B. McWilliams, et al. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, pages 724–732. 2016.

[12] Vaswani, A., N. Shazeer, N. Parmar, et al. Attention is all you need. In *NIPS*. 2017.

[13] Badrinarayanan, V., F. Galasso, R. Cipolla. Label propagation in video sequences. In *CVPR*, pages 3265–3272. IEEE, 2010.

[14] Vijayanarasimhan, S., K. Grauman. Active frame selection for label propagation in videos. In *ECCV*, pages 496–509. Springer, 2012.

[15] Avinash Ramakanth, S., R. Venkatesh Babu. Seamseg: Video object segmentation using patch seams. In *CVPR*, pages 376–383. 2014.

[16] Caelles, S., K.-K. Maninis, J. Pont-Tuset, et al. One-shot video object segmentation. In *CVPR*, pages 221–230. 2017.

[17] Xiao, H., J. Feng, G. Lin, et al. Monet: Deep motion exploitation for video object segmentation. In *CVPR*, pages 1140–1148. 2018.

[18] Voigtlaender, P., B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*. 2017.

[19] Perazzi, F., A. Khoreva, R. Benenson, et al. Learning video object segmentation from static images. In *CVPR*, pages 2663–2672. 2017.

[20] Luiten, J., P. Voigtlaender, B. Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *ACCV*, pages 565–580. 2018.

[21] Yang, L., Y. Wang, X. Xiong, et al. Efficient video object segmentation via network modulation. In *CVPR*, pages 6499–6507. 2018.

[22] Chen, Y., J. Pont-Tuset, A. Montes, et al. Blazingly fast video object segmentation with pixel-wise metric learning. In *CVPR*, pages 1189–1198. 2018.

[23] Hu, Y.-T., J.-B. Huang, A. G. Schwing. Videomatch: Matching based video object segmentation. In *ECCV*, pages 54–70. 2018.

[24] Wug Oh, S., J.-Y. Lee, K. Sunkavalli, et al. Fast video object segmentation by reference-guided mask propagation. In *CVPR*, pages 7376–7385. 2018.

[25] Bhat, G., F. J. Lawin, M. Danelljan, et al. Learning what to learn for video object segmentation. In *ECCV*. 2020.

[26] Wang, X., R. Girshick, A. Gupta, et al. Non-local neural networks. In *CVPR*, pages 7794–7803. 2018.

[27] Bahdanau, D., K. Cho, Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*. 2015.

[28] Devlin, J., M.-W. Chang, K. Lee, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171—-4186. 2019.

[29] Radford, A., J. Wu, R. Child, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[30] Synnaeve, G., Q. Xu, J. Kahn, et al. End-to-end asr: from supervised to semi-supervised learning with modern architectures. In *ICML Workshops*. 2020.

[31] Dosovitskiy, A., L. Beyer, A. Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*. 2021.

[32] Carion, N., F. Massa, G. Synnaeve, et al. End-to-end object detection with transformers. In *ECCV*, pages 213–229. Springer, 2020.

[33] Parmar, N., A. Vaswani, J. Uszkoreit, et al. Image transformer. In *ICCV*, pages 4055–4064. PMLR, 2018.

[34] Ba, J. L., J. R. Kiros, G. E. Hinton. Layer normalization. In *NIPS Workshops*. 2016.

[35] Hendrycks, D., K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[36] Lin, T.-Y., P. Dollár, R. Girshick, et al. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125. 2017.

[37] Wu, Y., K. He. Group normalization. In *ECCV*, pages 3–19. 2018.

[38] Everingham, M., L. Van Gool, C. K. Williams, et al. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

[39] Lin, T.-Y., M. Maire, S. Belongie, et al. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014.

[40] Cheng, M.-M., N. J. Mitra, X. Huang, et al. Global contrast based salient region detection. *TPAMI*, 37(3):569–582, 2014.

[41] Shi, J., Q. Yan, L. Xu, et al. Hierarchical image saliency detection on extended cssd. *TPAMI*, 38(4):717–729, 2015.

[42] Hariharan, B., P. Arbeláez, L. Bourdev, et al. Semantic contours from inverse detectors. In *ICCV*, pages 991–998. IEEE, 2011.

[43] Loshchilov, I., F. Hutter. Decoupled weight decay regularization. In *ICLR*. 2019.

[44] Ioffe, S., C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*. 2015.

[45] Johnander, J., M. Danelljan, E. Brissman, et al. A generative appearance model for end-to-end video object segmentation. In *CVPR*, pages 8953–8962. 2019.

[46] Voigtlaender, P., J. Luiten, B. Leibe. Boltvos: Box-level tracking for video object segmentation. *arXiv preprint arXiv:1904.04552*, 2019.

[47] Shaw, P., J. Uszkoreit, A. Vaswani. Self-attention with relative position representations. In *NAACL*, pages 464–468. 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] see the end of Sec. 1
   (b) Did you describe the limitations of your work? [Yes] we discuss the failure cases in Sec. 6, and demonstrate them in Fig. 3.

(c) Did you discuss any potential negative societal impacts of your work? [N/A] the main task, video object segmentation, is a valuable fundamental task in video understanding. Compared to previous works, we will not introduce additional negative societal impacts.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [N/A] the paper does not contain any theoretical assumptions.

(b) Did you include complete proofs of all theoretical results? [N/A] the paper does not contain any theoretical proofs.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] the instructions and details needed to reproduce the main results are supplied in Sec. 5 and the supplementary materials.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see Sec. 5 and the supplementary materials.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] we follow the usual format used in previous state-of-the-art methods [3, 4, 5, 7, 25] to report and compare the results.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Sec. 5 and the supplementary materials.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] we cite the creators or original papers of all the related code, data, and models used in this paper.

(b) Did you mention the license of the assets? [No] all the assets are free for research study and widely used in previous related works.

(c) Did you include any new assets either in the supplemental material or as a URL? [No] but the code of our proposed approach will be made publicly available as soon as the paper is accepted.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] all the datasets are publicly available, free for research study, and commonly used in previous related works.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] all the datasets are commonly used in previous research works.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] we didn't use crowdsourcing or conduct research with human subjects.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] we didn't use crowdsourcing or conduct research with human subjects.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] we didn't use crowdsourcing or conduct research with human subjects.