Efficient Video Instance Segmentation via Tracklet Query and Proposal

Jialian Wu1, Sudhir Yarram1, Hui Liang2, Tian Lan2
Junsong Yuan1, Jayan Eledath2

1State University of New York at Buffalo 2Amazon

https://jialianwu.com/projects/EfficientVIS.html

Abstract

Video Instance Segmentation (VIS) aims to simultaneously classify, segment, and track multiple object instances in videos. Recent clip-level VIS takes a short video clip as input each time showing stronger performance than frame-level VIS (tracking-by-segmentation), as more temporal context from multiple frames is utilized. Yet, most clip-level methods are neither end-to-end learnable nor real-time. These limitations are addressed by the recent VIS transformer (VisTR) [25] which performs VIS end-to-end within a clip. However, VisTR suffers from long training time due to its frame-wise dense attention. In addition, VisTR is not fully end-to-end learnable in multiple video clips as it requires a hand-crafted data association to link instance tracklets between successive clips. This paper proposes EfficientVIS, a fully end-to-end framework with efficient training and inference. At the core are tracklet query and tracklet proposal that associate and segment regions-of-interest (RoIs) across space and time by an iterative query-video interaction. We further propose a correspondence learning that makes tracklets linking between clips end-to-end learnable. Compared to VisTR, EfficientVIS requires $15 \times$ fewer training epochs while achieving state-of-the-art accuracy on the YouTube-VIS benchmark. Meanwhile, our method enables whole video instance segmentation in a single end-to-end pass without data association at all.

1. Introduction

Video Instance Segmentation (VIS) is a challenging video task recently introduced in [28]. It aims to predict a tracklet segmentation mask with a class label for every appeared object instance in a video as illustrated in Fig. 1. Existing methods typically solve the VIS problem at either frame-level or clip-level. The frame-level methods [10, 14, 16, 24, 26, 29] follow a tracking-by-segmentation paradigm, which first performs image instance segmentation and then links the current masks with history tracklets via data association as shown in Fig. 1 (a). This paradigm typically requires complex data association algorithms and limits temporal context, making it susceptible to object occlusions. By contrast, the clip-level methods [2, 3, 15, 25] jointly perform segmentation and tracking clip-by-clip. Within each clip, object information is propagated back and forth. Such a paradigm usually performs stronger than the frame-level methods thanks to larger temporal receptive field. However, most clip-level methods are not end-to-end and require elaborated inference that causes slow speed. These issues are addressed by the recent VIS transformer (VisTR) [25] which extends image object detector DETR [7] to the VIS task. VisTR generates VIS predictions within each video clip in one end-to-end pass, which greatly simplifies the clip-level paradigm and makes VIS within a clip end-to-end trainable.

However, the VIS transformer is confronted with two issues: (i) The convergence speed of VisTR is slow since the frame-wise dense attention weights in the transformer need long training epochs to search for properly attended regions among all video frame pixels. It is difficult for researchers to experiment with this algorithm as it requires long development cycles. (ii) When a video is too long to fit into GPU memory in one forward-pass, it has to be divided into multiple successive clips for sequential processing. In
In this case, VisTR becomes partially end-to-end. As shown in Fig. 1 (b), VisTR requires a hand-crafted data association to link tracklets between clips. Such a hand-crafted scheme is not only more complicated but also less effective than an end-to-end learnable association as evidenced in Table 1d.

In this paper, we propose a new clip-level VIS framework, coined as EfficientVIS. EfficientVIS delivers a fully end-to-end framework that can achieve fast convergence and inference with strong performance. Our work is inspired by the recent success of the query-based R-CNN architecture [10, 20] in image object detection. EfficientVIS extends the spirit of sparse object proposal [20] to the video domain to model the complex space-time interactions. Specifically, EfficientVIS uses tracklet query paired with tracklet proposal to represent an individual object instance in video. Tracklet query is latent embeddings that encode appearance information for a target instance, while tracklet proposal is a space-time tube that locates the target in the video. Our tracklet query collects the target information by interacting with video in a clip-by-clip fashion through a designed temporal dynamic convolution. This way enriches temporal object context that is an important cue for handling object occlusion and motion blur in videos. We also design a factorised temporo-spatial self-attention allowing tracklet queries to exchange information over not only space but also time. It enables one query to correlate a target across multiple frames so as to end-to-end generate target tracklet mask as a whole in a video clip. Compared to prior works, EfficientVIS enjoys three remarkable properties:

(i) Fast convergence: In EfficientVIS, tracklet queries interact with video CNN features only in the region of the space-time RoIs defined by the tracklet proposal. This is different from VisTR [25] that is interacting with all video pixels on the transformer [22] encoded features using dense attention. Our Rol-wise design drastically reduces video redundancies and therefore allows EfficientVIS for faster convergence than transformer as shown in Table 1g.

(ii) Fully end-to-end learnable framework: EfficientVIS goes beyond a short clip and is fully end-to-end learnable over the whole video. When there are multiple successive clips, one needs to link instance tracklets between clips. In contrast to prior clip-level works [3,18,23,25] that manually stitch the tracklets, we design a correspondence learning that enables tracklet query to be shared among clips for seamlessly associating a same instance. In other words, the tracklet query output from one clip is enabled to be fed into the next clip to associate and segment the same instance. Meanwhile, the query is dynamically updated in terms of the next clip content so as to achieve a continuous tracking for the future. Such a scheme makes EfficientVIS fully end-to-end, without any explicit data association for either inner-clip or inter-clip tracking as shown in Fig. 1 (c).

(iii) Tracking in low frame rate videos: Tracking object instances that have dramatic movements is a great challenge for motion-based trackers [18,23,31], as they suppose instances move smoothly over time. In contrast, our method retrieves a target instance in a frame conditioned on its query representation regardless of where it is in nearby frames. Thus, EfficientVIS is robust to dramatic object movements and can track instances in low frame rate videos as shown in Fig. 5 and Table 1h.

We summarize our major contributions as follows:

- **EfficientVIS is the first Rol-wise clip-level VIS framework that runs in real-time.** The Rol-wise design enables a fast convergence by drastically reducing video redundancies. Fully end-to-end learnable tracking and rich temporal context of the clip-by-clip workflow together bring a strong performance. EfficientVIS ResNet-50 achieves 37.9 AP on Youtube-VIS [28] in 36 FPS by training 33 epochs, which is 15× training epochs fewer and 2.3 AP higher than V isTR transformer.

- **EfficientVIS is the first fully end-to-end neural net for VIS.** Given a video as input despite its length, EfficientVIS directly produces VIS predictions without any data association or post-processing. We will demonstrate by diagnostic experiments that this fully end-to-end paradigm is not only simpler but also more effective than the previous partially/non end-to-end frameworks.

2. Related Works

**Frame-level VIS:** Most video instance segmentation methods work at the frame-level fashion, a.k.a. tracking-by-segmentation [6, 10, 11, 14, 16, 19, 24, 26, 29]. This paradigm produces instance segmentation frame-by-frame and achieves tracking by linking the current instance mask to the history tracklet. So it requires an explicit data association algorithm in either a hand-crafted or trainable way. Some attempts [11, 14, 26, 29] strive to exploit temporal context to improve VIS, yet their temporal context is limited to one or only a few frames ignoring fertile resources of videos. In contrast to the above methods, our method does not require data association algorithm at all and we take advantage of richer temporal context by performing VIS clip-by-clip.

**Clip-level VIS:** Recent clip-level works [2,3,15,25] demonstrate promising VIS accuracy by exploiting rich temporal knowledge from multiple frames of a video clip. This paradigm propagates object information back and forth in a clip, which can well handle object occlusion and motion blur. However, most clip-level methods have a complex inference process (e.g., extra mask refinement, box ensemble, etc) that makes them neither end-to-end learnable nor real-time in inference. To address these limitations, the VIS transformer (VisTR) [25] extends DETR [7] from images to videos, achieving efficient and end-to-end VIS within each clip. Nevertheless, VisTR suffers from slow convergence due to the frame-wise dense attention. Besides, if there are two or
more video clips, VisTR is not fully end-to-end and requires a hand-crafted data association to link tracklets between clips. In contrast to the above methods, EfficientVIS is fully end-to-end learnable without the need of data association for either inner- or inter-clips. Moreover, EfficientVIS performs VIS in an RoI-wise manner through an efficient query-video interaction, enabling fast convergence and inference.

3. EfficientVIS

EfficientVIS performs VIS clip-by-clip, where we present the main VIS architecture for inner-clip in Sec. 3.1 and the inter-clip tracklets correspondence learning in Sec. 3.3.

3.1. Main Architecture

EfficientVIS is an RoI-wise clip-level VIS framework. In each forward-pass, it takes as input a video clip \( \{ I_i \}_{i=1}^T \) and directly yields the VIS predictions, i.e. tracklet masks \( \{ m_i \}_{i=1}^N \) along with tracklet classifications, where \( I_i \in \mathbb{R}^{H_i \times W_i \times 3} \) and \( m_i \in \mathbb{R}^{T \times H_i \times W_i \times 1} \). \( T \) is the number of frames in the clip, and \( N \) is the number of unique object instances. \( H_i \times W_i \) is the frame spatial size. As detailed in Fig. 2, EfficientVIS starts with a CNN backbone that extracts a video base feature. Afterward, EfficientVIS iterates over the following query-video interaction \( M \) times: 1) Query interaction: tracklet queries self-interact and communicate in space and time by our factorised tempo-spatial self-attention so that a tracklet query consistently associates the same instance over time and different tracklet queries track different instances; 2) Video interaction: tracklet queries interact with video feature to collect the target instance information by our temporal dynamic convolution. At the end of each interaction, a couple of head networks are applied to update tracklet queries, proposals, masks, and classifications. Finally, EfficientVIS takes tracklet masks and classifications output from the last iteration as the VIS results.

Base Feature: We apply a standard CNN backbone frame-by-frame to the input video clip. The extracted features are then concatenated along the time dimension to form the base feature \( f \in \mathbb{R}^{T \times H \times W \times C} \), where \( C \) is the number of channels and \( H \times W \) is the spatial size of feature map. The base feature contains full semantic information of the clip.

Tracklet Query and Proposal: We use tracklet queries \( \{ q_i \}_{i=1}^N \) paired with tracklet proposals \( \{ b_i \}_{i=1}^N \) to represent every unique object instance throughout a video. Tracklet query \( q_i \) is embedding vectors that encode exclusive appearance information for the \( i \)-th instance, indicating the instance identity. Tracklet proposal is a space-time box tube \( b_i \in \mathbb{R}^{T \times 4} \), which tracks and locates the \( i \)-th instance across the entire video clip. \( b_i^t \in \mathbb{R}^4 \) determines the four corners of the box at time \( t \). Each tracklet query has \( T \) embeddings, i.e. \( q_i \in \mathbb{R}^{T \times C} \), where \( C \) is the channel number of an embedding. In this way, we enforce different embeddings to focus on different frames. This is more effective than using only one embedding as evidenced in Table 1f, because one instance at different time could have significant appearance changes. It is harsh to enforce a single embedding to encode various appearance patterns of an instance.

Factorised Temporo-Spatial Self-Attention (FTSA): The goal of FTSA is to achieve a query-target correspondence that makes each query associate to a unique target instance in the video. We take advantage of the idea of factorised temporo-spatial self-attention in the video transformer [1,4] which is originally used for encoding video backbone features. Here, we exploit it on instance tracklet queries to achieve an instance communication. Concretely, we separately perform temporal and spatial Multi-Head Self-Attention (MSA) [22] one after the other on tracklet queries as shown in Fig. 2. Let \( q_i^t \in \mathbb{R}^{1 \times C} \) denote the \( t \)-th embedding in the \( i \)-th instance tracklet query. The temporal self-attention is computed within the same tracklet query, across the \( T \) embedding pairs as:

\[
\{ q_i^t \}_{t=1}^T \leftarrow \text{MSA}(\{ q_i^t \}_{t=1}^T), \quad i = 1, \ldots, N.
\]  

The temporal self-attention allows the embeddings of one instance query from different frames to exchange the target
instance information, such that the embeddings can jointly associate the same instance over time. Therefore, instance association/tracking in each clip is implicitly achieved and end-to-end learned by our temporal self-attention. The spatial self-attention is then computed at the same frame, across the N embedding pairs from different tracklet queries as:

\[ \{q^t_i\}_{i=1}^N \leftarrow \text{MSA}(\{q^t_i\}_{i=1}^N), \quad t = 1, \ldots, T. \]  

(2)

The spatial self-attention allows each query to acquire object context from other queries in each frame, reasoning about the relations among different object instances. We observe in experiments that the order of temporal and spatial self-attention does not cause a noticeable performance difference.

Joint tempo-spatial self-attention that computes MSA across all \( T \times N \) embedding pairs for once is another alternative for a query communication. Compared to this scheme, our FTSA saves more computation and is more effective as we will demonstrate in Table 1a.

**Temporal Dynamic Convolution (TDC):** The aim of the TDC is to collect the target instance information from the video clip. Specifically, we generate a dynamic convolutional filter [13, 21] conditioned on tracklet query embedding. We then use it to perform convolution on an RoI region of the base feature specified by the corresponding tracklet proposal. Different from the still-image dynamic convolution [10, 20, 21], we perform a 3D dynamic convolution in order to collect temporal instance context from nearby frames as well. Let \( w^t_i \) denote the dynamic convolutional filter generated from tracklet query embedding \( q^t_i \). The TDC is computed as:

\[ \alpha^t_i = \sum_{t'=t-1}^{t+1} a_{i,(t,t')} \circ \text{conv2d}(w^t_i, \phi(f^t', b^t')). \]  

(3)

\( \phi \) denotes RoI align [12] whose output spatial size is \( H_r \times W_r \) and \( \circ \) is element-wise product. \( a_{i,(t,t')} \in \mathbb{R}^{H_r \times W_r} \) is a simple adaptive weight that measures the similarity of \( \text{conv2d}(\cdot) \) outputs between time step \( t \) and \( t' \). Similar to [27, 32], the adaptive weight is calculated by cosine similarity and softmax such that \( \sum_{t'=t-1}^{t+1} \alpha^t_i = 1 \). \( \{\alpha^t_i\}_{t=1}^T \in \mathbb{R}^{T \times H_r \times W_r \times C} \) is the tracklet feature of the \( i \)-th instance. Eq. 3 differs from regular 3D convolution, where we share the same filter over the time dimension.

Since the dynamic filter \( w_i \) is generated from \( q_i \), \( w_i \) has distinct cues and appearance semantics of the \( i \)-th instance. The TDC exploits \( w_i \) to exclusively collect the \( i \)-th instance information from the video base feature yet filter out irrelevant information. Therefore, tracklet feature \( \alpha_i \) shall be highly activated by \( w_i \) in the region where the \( i \)-th instance appears but will be suppressed in the region of uncorrelated instances or background clutter as evidenced in Fig. 4.

**Head Networks:** For each instance \( i \), we employ lightweight head networks on its tracklet feature \( \alpha_i \) to output its VIS predictions and renewed tracklet query and proposal. Since tracklet feature \( \alpha_i \) exclusively carries the \( i \)-th instance information across the clip, these outputs can be readily derived by applying regular convolutions or fully connected layers (FCs). Specifically, we apply convolutions to segment tracklet mask, and FCs to classify tracklet, regress tracklet proposal, and update tracklet query. The updated query and proposal are fed into the next iteration. The head network architectures are similar to [10, 12, 20]. In a nutshell, our tracklet (proposal) for a video clip is directly generated as a whole determined by tracklet query. So we do not need explicit data association to link instances/boxes across frames in a clip as frame-level VIS methods [10, 11, 14, 16, 19, 24, 29].

**Initial Tracklet Query and Proposal Details:** As [10, 20], the initial tracklet query and proposal at the input of the first iteration are model parameters learned from training. Each initial query has one embedding, i.e. \( q^*_1 \in \mathbb{R}^{1 \times C} \). As shown in Fig. 2, we repeat it \( T \) times over the time dimension before starting the first iteration\(^1\). Besides, we find in experiments that all the learned initial proposals tend to be the frame size which covers the whole scene. We think this is reasonable because it is easier for later stages to regress the proposal to the target region by first taking a glance at the whole image.

### 3.2. Training

For each video clip, EfficientVIS is trained by the one-to-one matching loss [7, 10, 20, 25], which first obtains a one-to-one assignment between predictions and ground truths by bipartite matching and then computes the training loss in terms of the assignment. Let \( \{y_j\}_{j=1}^N \) denote the tracklet predictions and \( \{\hat{y}_j\}_{j=1}^C \) denote the tracklet ground truths, where \( G (\leq N) \) is the total number of ground truths. We denote \( \sigma \) as a \( G \)-elements permutation of \( \{1, \ldots, N\} \). The bipartite matching between \( y \) and \( \hat{y} \) is found by searching for a \( \sigma \) with the lowest cost as:

\[ \hat{\sigma} = \arg \min_{\sigma} \sum_{j=1}^{G} L_{\text{match}}(y_{\sigma(j)}, \hat{y}_j). \]  

(4)

The optimal assignment \( \hat{\sigma} \) is obtained by the Hungarian algorithm. Each instance tracklet has three types of annotation \( \hat{y}_j = (\hat{c}_j, \hat{b}_j, \hat{m}_j) \), where \( \hat{c}_j \in \mathbb{R}^T \) is the class label including background class \( \emptyset \), \( \hat{c}_j = \emptyset \) indicates the instance disappears at time step \( t \). Let \( p_{\sigma(j)}(\hat{c}_j) \) be the predicted classification score of the \( \sigma(j) \)-th tracklet for the class \( \hat{c}_j \). The matching cost is computed by \( L_{\text{match}} = \sum_{t=1}^{T} -p_{\sigma(j)}(\hat{c}_j^t) + \mathbb{1}_{[\hat{c}_j^t = \emptyset]} (L_{\text{box}}(\hat{b}_j^t) - L_{\text{maskdet}}(\hat{m}_j^t)) \), where \( b \) and \( m \) are the predicted tracklet proposal and mask respectively. \( L_{\text{box}} \) is calculated by box IoU and L1 distance as [7]. After finding the optimal assignment \( \hat{\sigma} \) that one-to-one matches between tracklet predictions and ground truths, the training loss for the video clip is computed as:

\[^1\text{We repeat the parameter} q^* \text{ times rather than directly setting the parameter to be} q^* \in \mathbb{R}^{T \times C}, \text{because it gives an initial prior that all the} T \text{ embeddings shall belong to a same instance.} \]
Figure 3. (a) Correspondence learning. (b) Fully end-to-end inference in multiple clips.

$$\mathcal{L}_{\text{clip}}(\hat{\sigma}) = \sum_{j=1}^{G} \sum_{t=1}^{T} \frac{-p_{\hat{\sigma}(j)}( \hat{c}_t^j )}{L_{\text{CE}}(m_{\hat{\sigma}(j)}^t, \hat{m}_t^j)}$$  

$$+ 1_{\{\hat{c}_t^j \neq \emptyset\}} \left( \mathcal{L}_{\text{box}}(b_{\hat{\sigma}(j)}^j, \hat{b}_t^j) + \mathcal{L}_{\text{dice}}(m_{\hat{\sigma}(j)}^t, \hat{m}_t^j) \right)$$  

where $L_{\text{CE}}$ is the binary cross-entropy loss and $L_{\text{dice}}$ is the dice loss [17]. All the losses and balancing weights are similar to [7, 10, 20, 25], except that we add a $L_{\text{CE}}$ to enforce predicted mask to be zero if the instance disappears. Eq. 5 is averaged by the total number of entries. For those unmatched tracklet predictions, we only impose classification loss to supervise them to be background class.

### 3.3. Correspondence Learning

If a long video cannot fit into GPU memory in one forward-pass, it needs to be split into multiple successive clips for sequential processing. To stitch tracklets between two clips, prior works [3, 18, 23, 25] turn to hand-crafted data association. However, human-designed rules are cumbersome and not end-to-end learnable. Therefore, we design a correspondence learning that makes the tracklet linking between clips end-to-end learnable without data association, which is not only simpler but also more effective.

**Correspondence learning**: As shown in Fig. 3, every two clips from the input video are paired\(^2\) in training. For the first clip of a pair, the initial tracklet queries are the model parameters $q'$ as described in Sec. 3.1, and the training procedure is the same as Sec. 3.2, i.e., finding the optimal assignment $\hat{\sigma}$ and backpropagating $\mathcal{L}_{\text{clip}}(\hat{\sigma})$. For the initial queries of the second clip, we use the output tracklet queries from the first clip averaged along the time dimension. The average operation can extract a comprehensive abstract for the instance representation. As for the assignment of the second clip, we do not perform bipartite matching. Instead, we use the same assignment $\hat{\sigma}$ that is already obtained by the first clip to train the second clip. Such a training manner enforces the tracklet queries output from one clip to segment and correspond to the same object instances in the next clip. In this way, tracklet query is able to serve as a generic video-level representation to seamlessly associate and correlate an instance across a whole video instead of only a clip.

\(^2\)The order of the two clips in a pair is randomized.

### 4. Experiments

#### 4.1. Experimental Setup

**Datasets**: Our experiments are conducted on the YouTube-VIS [28] benchmark in two versions. YouTube-VIS 2019 is the first dataset for video instance segmentation, which contains 2,883 videos labeled at 6 FPS, 131K instance masks, and 40 object classes. YouTube-VIS 2021 is an increased version that contains 3859 videos.

**Evaluation Metric**: The video-level average precision (AP) and average recall (AR) are the metrics. Different from the image domain, the video intersection over union (IoU) is computed between the predicted tracklet mask and ground truth tracklet mask. So, in order to achieve high performance, a model needs to not only correctly classify and segment the target instance but also accurately track the target over time.

**Architecture Settings**: Following [25], the default video clip length and the number of tracklet queries are set to $T = 36$ and $N = 10$, respectively. Following [10, 20, 25], the number of iterations is $M = 6$ and the default CNN backbone is ResNet-50. We use the above default settings for all experiments unless otherwise specified. Image frame size and pretrained model are the same as [6, 10, 25, 29].

**Training**: EfficientVIS is trained by AdamW with an initial learning rate of $2.5 \times 10^{-5}$. We train the model for 33 epochs where the learning rate is dropped by a factor of 10 at the 27-th epoch. For example, the model can be trained in 12 hours with 4 RTX 3090 GPUs on YouTube-VIS 2019. The correspondence learning is enabled if the maximum frame number of videos in a dataset is larger than $T$. No training data augmentation is used unless otherwise specified.

**Fully end-to-end Inference**: EfficientVIS does not include any data association or post-processing. **Reason**: Target tracklet **within** a clip is generated as a whole in a single forward-pass thanks to the implicit association of our FTSA (Sec. 3.1). Target tracklets **between** two clips are automatically correlated owing to our correspondence learning (Sec. 3.3). The one-to-one matching loss (Sec. 3.2) further enables our method to get rid of post-processing like NMS.
### 4.2. Ablation Studies

**Time Disentangled Query:** As described in Sec. 3.1, our tracklet query is designed to be disentangled in time, *i.e.* each query contains $T$ embeddings instead of one embedding. As shown in Table 1f, we compare it with the time shared query scheme that only uses one single embedding for each tracklet query. Our time disentangled query achieves 1.5 AP higher than the time shared one. We argue that this is because objects in different time may have dramatic appearance changes like heavy occlusion or motion blur. It is more reasonable to use multiple embeddings to separately encode different appearance patterns of an instance.

**Factorised Temporo-Spatial Self-Attention:** To evaluate our factorised temporo-spatial self-attention (FTSA), we experiment with different self-attention schemes as shown in Table 1a. “S” indicates the spatial self-attention that only performs multi-head self-attention (MSA) within the same frame across the embeddings of different tracklet queries. “T” denotes the temporal self-attention that only performs MSA within the same tracklet query across the embeddings of different time. “Joint T-S” is the joint temporo-spatial self-attention that performs MSA across all $T \times N$ embeddings as described in Sec. 3.1. We see from the table that the FTSA and “Joint T-S” achieve better performance than using either “S” or “T” standalone. This is because the temporal self-attention can let a query in different time exchange information so as to associate the same instance, while the spatial self-attention models different instances relations in space. Such a query communication in both space and time is necessary for the VIS task. Compared to “Joint T-S”, our FTSA yields better VIS results. We think the reason is that all attendees in FTSA are strongly related to each other. For example, the embeddings in temporal self-attention are all from the same tracklet query, while those in spatial self-attention are all from the same frame. However, by mixing up all $T \times N$ embeddings in a single self-attention pass, much less relevant information from other tracklets at far temporal positions is directly involved during “Joint T-S”.

**Temporal Dynamic Convolution:** To assess our temporal dynamic convolution (TDC), we compare it with the still-image dynamic convolution that performs 2D convolution on the current frame only. As shown in Table 1b, our temporal dynamic convolution improves 1 AP over the still-image version. It suggests that the temporal object context from nearby frames is also an informative cue. Moreover, we visualize the tracklet feature. As shown in Fig. 4, tracklet feature is highly activated by our TDC in the region where the target instance appears. We also see that tracklet feature is suppressed when the target instance disappears, even if its tracklet proposal has drifted to background clutter or uncorrelated instance. This demonstrates the dynamic filter...
conditioned on tracklet query is target-specific, which exclusively collects the query’s target information from the video base feature yet discards irrelevant information.

**Video Clip Length:** As shown in Table 1c, we experiment with different video clip lengths. EfficientVIS improves 1.7 AP by increasing the number of frames from 9 to 36. It is reasonable because more frames can provide richer temporal context which is important to video tasks. Moreover, since our method runs clip-by-clip in 36 FPS as shown in Table 2, it can be regarded as a near-online fashion with a delay of \( \frac{T}{36} \). For example, there is a 0.25s delay when \( T = 9 \).

**Correspondence Learning:** To validate the effectiveness of our correspondence learning, we train an EfficientVIS without the correspondence learning. Concretely, we treat every clip as an individual video and do not use tracklet queries from other clips as input during training. In inference, we keep our fully end-to-end paradigm, i.e., the output tracklet queries from one clip are used as the initial queries of the next clip. As shown in Table 1e, the performance significantly drops if EfficientVIS is trained without the correspondence learning. It suggests that our correspondence learning is the key to enabling tracklet query to be a video-level instance representation that can be shared among different video clips.

**Fully End-to-end Learnable Framework:** Thanks to the correspondence learning, EfficientVIS fully end-to-end performs VIS by taking tracklet queries from one clip as the initial queries of the next clip. To evaluate this fully end-to-end framework, we compare it with two partially end-to-end schemes, “hand-craft” and VisTR [25] in Table 1d. These two methods perform VIS end-to-end within each clip but require a hand-crafted data association to link tracklets between clips. For the “hand-craft” scheme, all the model architectures keep the same as EfficientVIS except that tracklets between two clips are linked by a human-designed rule: 1) We first construct an affinity matrix by taking into account box IoU, box L1 distance, mask IoU, and class label matching like prior tracking works [26, 31]; 2) We then solve the tracklet association by the Hungarian algorithm. As shown in Table 1d, our fully end-to-end scheme performs better than the partially end-to-end framework in different \( T \) settings while greatly simplifying the previous VIS paradigms. The reason for this performance gap is that the tracklet linking between clips in hand-crafted scheme tends to be sub-optimal, while that in our fully end-to-end framework is learned and optimized by ground truths during training.

**Tracking in Low Frame Rate Videos:** Tracking object instances in low frame rate videos is a hard problem, where motion-based trackers usually fail. Motion-based trackers suppose instances move smoothly over time and impose spatial movement constraints to prevent trackers from associating very distant instances. In contrast, our tracking is determined by instance appearance rather than motion, as we retrieve a target instance solely conditioned on its query representation regardless of the target spatial distance among different frames. Therefore, our tracking is not limited by dramatic instance movements. To demonstrate this property, we downsample the frame rate of the YouTube-VIS dataset to 1.5 FPS and test our method. As shown in Table 1h, EfficientVIS successfully maintains its VIS performance in the low frame rate video setting. As shown in Fig. 5, we also visualize the results of EfficientVIS on a very low frame rate video whose frame is sampled every 5 seconds, i.e. 0.2 FPS.

**Fast Convergence:** In our method, we crop video using tracklet proposal and collect target information from the proposal. Compared to VIS transformer (VisTR) [25] that uses frame-wise dense attention, this RoI-wise design eliminates much background clutter and redundancies in videos and enforces EfficientVIS to focus more on informative regions, making our convergence \( 15 \times \) faster as shown in Table 1g. This RoI-wise pipeline also leads to better accuracy than VisTR. This is because tracklet proposal can avoid regions outside the target being segmented (second figure in appendix), which results in more precise masks and significantly higher AP75 as evidenced in Table 1g.

### 4.3. Comparison to State of the Art

**YouTube-VIS 2019:** In Table 2, we compare EfficientVIS with the real-time state-of-the-art VIS models. EfficientVIS is the only fully end-to-end framework while achieving superior accuracy. We attribute the strong performance to two main aspects: 1) Our object tracking is learned via the end-to-end framework; 2) Our clip-by-clip processing and temporal dynamic convolution enrich temporal object context that is helpful for handling occlusion or motion blur. As shown in Table 2, we achieve notably higher recall (AR)
than other competitors. We think the reason for the high recall is that we recognize more heavily occluded or blurred objects, which are usually missed by common methods.

For a fair comparison, Table 2 only lists real-time methods using single model without extra data. There are two works with 40+ AP, MaskProp [3] (<5.6 FPS) and Propose-Reduce [15] (1.8 FPS), which however are not real-time and adopt multiple models or extra training data. MaskProp uses DCN backbone [9], HTC detector [8], extra High-Resolution Mask Refinement network, etc. Propose-Reduce is trained with additional DAVIS-UVOS [5] and COCO pseudo videos datasets. These elaborated implementations are beyond the scope of our work, as our main goal is to present a simple end-to-end model with real-time inference and fast training.

**YouTube-VIS 2021**: We experiment with EfficientVIS on YouTube-VIS 2021 in Table 3. EfficientVIS achieves state-of-the-art AP without bells and whistles. Similar to the findings on YouTube-VIS 2019, EfficientVIS significantly improves AR over other state of the arts.

5. Conclusion

This work presents a new VIS model, EfficientVIS, which simultaneously classifies, segments, and tracks multiple object instances in a single end-to-end pass and clip-by-clip fashion. EfficientVIS adopts tracklet query and proposal to respectively represent instance appearance and position in videos. An efficient query-video interaction is proposed for associating and segmenting instances in each clip. A correspondence learning is designed to correlate tracklets between clips without data association. The above designs enable a fully end-to-end framework achieving state-of-the-art VIS accuracy with fast training and inference.

Acknowledgement. This paper is supported in part by a gift grant from Amazon Go and National Science Foundation Grant CNS1951952.

Table 2. Comparison with real-time state-of-the-art methods on the YouTube-VIS 2019 val set.

<table>
<thead>
<tr>
<th>Method</th>
<th>Publication</th>
<th>Augmentations</th>
<th>Backbone</th>
<th>FPS</th>
<th>AP</th>
<th>AP50</th>
<th>AP75</th>
<th>AR1</th>
<th>AR10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaskTrack R-CNN [28]</td>
<td>ICCV’19</td>
<td>×</td>
<td>ResNet-50</td>
<td>33</td>
<td>30.3</td>
<td>31.1</td>
<td>33.0</td>
<td>35.2</td>
<td>35.5</td>
</tr>
<tr>
<td>SipMask [6]</td>
<td>ECCV’20</td>
<td>×</td>
<td>ResNet-50</td>
<td>34</td>
<td>32.5</td>
<td>33.3</td>
<td>36.3</td>
<td>37.2</td>
<td>38.9</td>
</tr>
<tr>
<td>CompFeat [11]</td>
<td>AAAI’21</td>
<td>×</td>
<td>ResNet-50</td>
<td>&lt;33</td>
<td>35.3</td>
<td>36.0</td>
<td>38.6</td>
<td>33.1</td>
<td>40.3</td>
</tr>
<tr>
<td>TraDeS [26]</td>
<td>CVPR’21</td>
<td>×</td>
<td>ResNet-50</td>
<td>26</td>
<td>32.6</td>
<td>32.8</td>
<td>39.1</td>
<td>31.6</td>
<td>36.6</td>
</tr>
<tr>
<td>QueryInst [10]</td>
<td>ICCV’21</td>
<td>×</td>
<td>ResNet-50</td>
<td>32</td>
<td>34.6</td>
<td>35.8</td>
<td>36.5</td>
<td>35.4</td>
<td>42.4</td>
</tr>
<tr>
<td>CrossVIS [29]</td>
<td>ICCV’21</td>
<td>×</td>
<td>ResNet-50</td>
<td>40</td>
<td>34.8</td>
<td>34.6</td>
<td>37.9</td>
<td>34.0</td>
<td>39.0</td>
</tr>
<tr>
<td>ViSTG [24]</td>
<td>ICCV’21</td>
<td>×</td>
<td>ResNet-50</td>
<td>22</td>
<td>35.2</td>
<td>35.7</td>
<td>38.0</td>
<td>33.6</td>
<td>38.5</td>
</tr>
<tr>
<td><strong>EfficientVIS (Ours)</strong></td>
<td>CVPR’22</td>
<td>×</td>
<td>ResNet-50</td>
<td>36</td>
<td>37.0</td>
<td>59.6</td>
<td>40.0</td>
<td>39.3</td>
<td>46.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Publication</th>
<th>Augmentations</th>
<th>Backbone</th>
<th>FPS</th>
<th>AP</th>
<th>AP50</th>
<th>AP75</th>
<th>AR1</th>
<th>AR10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaskTrack R-CNN [28]</td>
<td>ICCV’19</td>
<td>×</td>
<td>ResNet-101</td>
<td>29</td>
<td>33.5</td>
<td>31.1</td>
<td>33.1</td>
<td>31.1</td>
<td>39.2</td>
</tr>
<tr>
<td>SRNet [30]</td>
<td>ACMMM’21</td>
<td>×</td>
<td>ResNet-101</td>
<td>35</td>
<td>32.3</td>
<td>32.3</td>
<td>32.3</td>
<td>32.3</td>
<td>40.1</td>
</tr>
<tr>
<td>CrossVIS [29]</td>
<td>ICCV’21</td>
<td>×</td>
<td>ResNet-101</td>
<td>36</td>
<td>36.6</td>
<td>36.7</td>
<td>39.7</td>
<td>36.0</td>
<td>42.0</td>
</tr>
<tr>
<td><strong>EfficientVIS (Ours)</strong></td>
<td>CVPR’22</td>
<td>×</td>
<td>ResNet-101</td>
<td>32</td>
<td>38.7</td>
<td>61.3</td>
<td>44.0</td>
<td>40.6</td>
<td>47.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Publication</th>
<th>Augmentations</th>
<th>Backbone</th>
<th>FPS</th>
<th>AP</th>
<th>AP50</th>
<th>AP75</th>
<th>AR1</th>
<th>AR10</th>
</tr>
</thead>
<tbody>
<tr>
<td>SipMask [6]</td>
<td>ECCV’20</td>
<td>×</td>
<td>ResNet-101</td>
<td>24</td>
<td>35.8</td>
<td>39.0</td>
<td>39.0</td>
<td>35.4</td>
<td>42.4</td>
</tr>
<tr>
<td>STMask [14]</td>
<td>CVPR’21</td>
<td>×</td>
<td>ResNet-101</td>
<td>23</td>
<td>36.8</td>
<td>38.0</td>
<td>38.0</td>
<td>34.8</td>
<td>41.8</td>
</tr>
<tr>
<td>SG-Net [16]</td>
<td>CVPR’21</td>
<td>×</td>
<td>ResNet-101</td>
<td>20</td>
<td>36.3</td>
<td>39.6</td>
<td>35.9</td>
<td>39.6</td>
<td>43.0</td>
</tr>
<tr>
<td>VisTR [25]</td>
<td>CVPR’21</td>
<td>×</td>
<td>ResNet-101</td>
<td>28</td>
<td>38.6</td>
<td>61.3</td>
<td>42.3</td>
<td>37.6</td>
<td>44.2</td>
</tr>
<tr>
<td><strong>EfficientVIS (Ours)</strong></td>
<td>CVPR’22</td>
<td>×</td>
<td>ResNet-101</td>
<td>32</td>
<td>39.8</td>
<td>61.8</td>
<td>44.7</td>
<td>42.1</td>
<td>49.8</td>
</tr>
</tbody>
</table>

Table 3. Comparison with real-time state-of-the-art methods on the YouTube-VIS 2021 val set. × denotes multi-scale training.
References


