Video Instance Segmentation using Inter-Frame Communication Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose a novel end-to-end solution for video instance segmentation (VIS) based on transformers. Recently, the *per-clip* pipeline shows superior performance over *per-frame* methods leveraging richer information from multiple frames. However, previous per-clip models require heavy computation and memory usage to achieve frame-to-frame communications, limiting practicality. In this work, we propose Inter-frame Communication Transformers (IFC), which significantly reduces the overhead for information-passing between frames by efficiently encoding the context within the input clip. Specifically, we propose to utilize concise *memory tokens* as a means of conveying information as well as summarizing each frame scene. The features of each frame are enriched and correlated with other frames through exchange of information between the precisely encoded memory tokens. We validate our method on the latest benchmark sets and achieved the state-of-the-art performance (AP 44.6 on YouTube-VIS 2019 val set using the offline inference) while having a considerably fast runtime (89.4 FPS). Our method can also be applied to near-online inference for processing a video in real-time with only a small delay. The code will be made available.

1 Introduction

With the growing interest toward the video domain in computer vision, the task of video instance segmentation (VIS) is emerging [1]. Most of the current approaches [1, 2, 3, 4] extend image instance segmentation models [5, 6, 7, 8] and take frame-wise inputs. These per-frame methods extend the concept of temporal tracking by matching frame-wise predictions of high similarities. The models can be easily customized to real-world applications as they run in an online [9] fashion, but they show limitation in dealing with occlusions and motion blur that are common in videos.

On the contrary, per-clip models are designed to overcome such challenges by incorporating multiple frames while sacrificing the efficiency. Previous per-clip approaches [10, 11, 12] aggregate information within a clip to generate instance-specific features. As the features are generated per instance, the number of instances in addition to the number of frames has a significant impact on the overall computation. Recently proposed VisTR [11] adapted DETR [13] to the VIS task and reduced the inference time by inserting the entire video, not a clip, to its offline end-to-end network. However, its full self-attention transformers [14] over the space-time inputs involve explosive computations and memories. In this work, we raise the following question: can a per-clip method be efficient while attaining great accuracy?

To achieve our goal, we introduce Inter-frame Communication Transformers (IFC) to greatly reduce the computations of the full space-time transformers. Similar to recent works [15, 16] that alleviate the explosive computational growth inherent in attention-based models [17, 14], IFC takes a decomposition strategy utilizing two transformers. The first transformer (Encode-Receive, $\mathcal{E}$) encodes

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

each frame independently. To exchange the information between frames, the second transformer (Gather-Communicate, $\mathcal{G}$) executes attention between a small number of memory tokens that hold concise information of the clip. The memory tokens are utilized to store the overall context of the clip, for example *"a hand over a lizard"* in Fig. 1. The concise information assists detecting the lizard that is largely occluded by the hand in the first frame, without employing an expensive pixel-level attention over space and time. The memory tokens are only in charge of the communications between frames, and the features of each frame are enriched and correlated through the memory tokens.

We further reduce overheads while taking advantages of per-clip pipelines by concisely representing each instance with a unique convolutional weight [7]. Despite the changes of appearances at different frames, the instances of the same identity share commonalities because the frames are originated from the same source video. Therefore, we can effectively capture instance-specific characteristics in a clip with dynamically generated convolutional weights. In companion with the segmentation, we track instances by uniformly applying the weights to all frames in a clip. Moreover, all executions of our spatial decoder are instance-agnostic except for the final layer which applies instance-specific weights. Accordingly, our model is highly efficient and also suitable for scenes with numerous instances.

In addition to the efficient modeling, we provide optimizations and an instance tracking algorithm that are designed to be VIS-centric. By the definition of $\text{AP}^{\texttt{VIS}}$, the VIS task [1] aims to maximize the objective similarity: space-time mask IoU. Inspired by previous works [13, 18, 19], our model is optimized to maximize the similarity between bipartitely matched pairs of ground truth masks and predicted masks. Furthermore, we again adopt the similarity maximization for tracking instances of same identities, which effectively links predicted space-time masks using bipartite matching. As both of our training and inference algorithm are fundamentally designed to address the key challenge of VIS task, our method attains an outstanding accuracy.

From these improvements, IFC sets the new state-of-the-art by using ResNet-50: 42.8% AP and more surprisingly, in 107.1 fps. Furthermore, our model also shows great speed-accuracy balance under near-online setting, which leads to a huge practicality. We believe that our model can be a powerful baseline for video instance segmentation approaches that follow the per-clip execution.

2 Related Work

Video instance segmentation The VIS task [1] extends the concept of tracking to the image instance segmentation task. The early solutions [1, 2] follow per-frame pipeline, which utilize additional tracking head to the models that are mainly designed to solve image instance segmentation. More advanced algorithms that are recently proposed [4, 3] take video characteristics into consideration, which result in improved performance.

Per-clip models [12, 10, 11] dedicate computations to extract information from multiple frames for higher accuracy. By exploiting multiple frames, per-clip models can effectively handle typical challenges in video, *i.e.*, motion blurs and occlusions. Our model is designed to be highly efficient while following the per-clip pipeline, which leads to fast and accurate predictions.

Transformers Recently, transformers [14] are greatly impacting many tasks in computer vision. After the huge success of DETR [13], which has brought a new paradigm to the object detection task, numerous vision tasks are incorporating transformers [20, 21] in place of CNNs. For classification tasks in both NLP and computer vision, many adopt an extra classification token to the input of transformers [22, 20]. All the input tokens affect each other as the encoders are mainly composed of the self-attention, thus the classification token can be used to determine the class of the overall input. MaX-DeepLab [19] adopted the concept of memory and proposed a novel dual-path transformer for the panoptic segmentation task [23]. By making use of numerous memory tokens similar to the previous classification token, MaX-DeepLab integrates the transformer and the CNN by making both feedback itself and the other.

We further utilize the concept of the memory tokens to the videos. Using Inter-frame Communication Transformers, each frame runs independently while sharing their information with interim communications. The communications lead to higher accuracy while the execution independence between frames accelerates the inference.

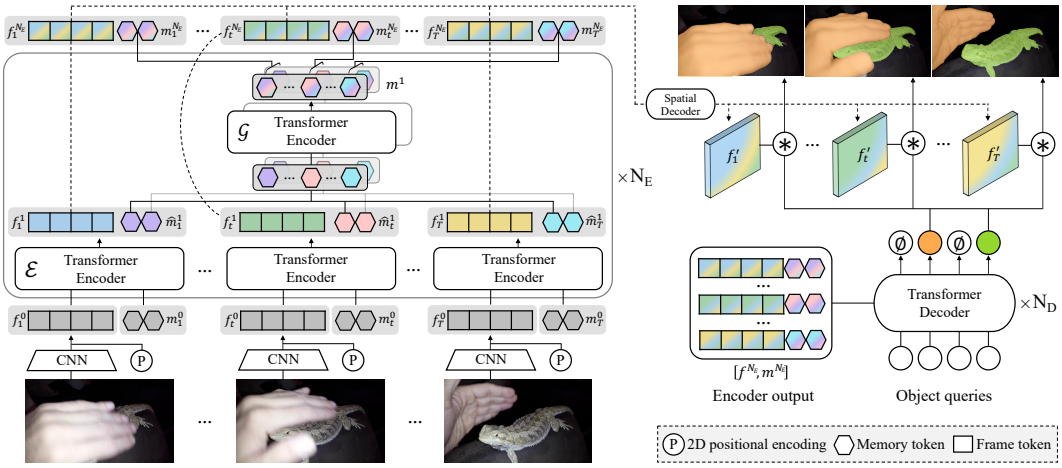

Figure 1: Overview of IFC framework. Our transformer encoder block has two components: 1) Encode-Receive ($\mathcal{E}$) simultaneously encodes frame tokens and memory tokens. 2) Only memory tokens pass Gather-Communicate ($\mathcal{G}$) to perform communications between frames. The outputs from the stack of N_E encoder blocks goes into two modules, spatial decoder and transformer decoder, to generate segmentation masks.

3 Method

The proposed method follows a per-clip pipeline which takes a video clip as input and outputs clip-level results. We also introduce Inter-frame Communication Transformers, which can effectively share frame-wise information within a clip with a high efficiency.

3.1 Model architecture

Inspired by DETR [13], our network consists of a CNN backbone and transformer encoder-decoder layers (Fig. 1). The input clip is first independently embedded into a feature map through the backbone. Then, the embedded clip passes through our *inter-frame communication* encoder blocks that enrich the feature map by allowing information exchange between frames. Next, a set of transformer decoder layers that take the encoder outputs and object queries as inputs predict unique convolutional weights for each instance in the clip. Finally, the masks for each instance across the clip are computed in one shot by convolving the encoded feature map with the unique convolutional weight.

Backbone Given an input clip $\{x_i\}_{i=1}^{T} \in \mathbb{R}^{T\times H_0\times W_0\times 3}$, composed of T frames with 3 color channels, the CNN backbone processes the input clip frame-by-frame. As the result, the clip is encoded into a set of low-resolution features, $\{f_i^0\}_{i=1}^{T} \in \mathbb{R}^{T\times H\times W\times C}$, where C is the number of channels and $H, W = \frac{H_0}{32}, \frac{W_0}{32}$.

Inter-Frame Communication Encoder Given an image, humans can effortlessly summarize the scene with only a few words. Also, frames from a same video share a lot of commonalities, the difference between them is sufficiently summarized and communicated even with a small bandwidth. Based on this hypothesis, we propose an inter-frame communication encoder to make the computation to be mostly frame-wise independent with some communications between frames. Specifically, we adopt memory tokens for both summarizing per-frame scenes and the means of communications.

Our encoder blocks are composed of two phases of separate transformers: Encode-Receive ($\mathcal{E}$) and Gather-Communicate ($\mathcal{G}$). Both Encode-Receive and Gather-Communicate follow the typical transformer encoder architecture [14], which consists of an addition of fixed positional encoding, a multi-head self-attention module, and a feed forward network.

Encode-Receive operates in a per-frame manner, taking a frame-level feature map and corresponding memory tokens. Passing through Encode-Receive, we expect two functionalities: (1) image features encode per-frame information to the memory tokens, and (2) image features receive information of different frames that are gathered in the memory tokens. Gather-Communicate operates across frames

Table 1: Complexity comparison. Various transformer encoders for space-time input. As the overall FLOPs can vary by the number of detected instances, listed values are measured only at the encoders.

Communication Type	Complexity per Layer	FLOPs (G)[1]			
		360×640		720×1280	
		T=5	T=36	T=5	T=36
No Comm	$\mathcal{O}(C^2THW + CT(HW)^2)$	5.17	37.23	24.62	177.29
Full THW	$\mathcal{O}(C^2THW + C(THW)^2)$	6.94	148.70	50.63	1815.38
Decompose T-HW	$\mathcal{O}(C^2THW + CT(HW)^2 + CT^2HW)$	8.33	60.24	36.73	265.50
IFC ($M = 8$)	$\mathcal{O}(C^2THW + CT(HW)^2)$	5.52	39.73	25.05	180.39

to form a clip-level knowledge. It takes the memory tokens from each frame as inputs and performs communications between frames. Alternating two phases through multiple layers, the encoder can efficiently learn consensus representations across frames.

In more detail, given the frame embedding $\{f_i^0\}_{i=1}^T$, we spatially flatten each feature $\mathbb{R}^{H\times W\times C} \rightarrow \mathbb{R}^{HW\times C}$. The initial memory tokens m^0 of size M are copied per frame and concatenated to each frame feature as follows:

$$[f_t^0, m_t^0] \in \mathbb{R}^{(HW+M)\times C}, \qquad t \in \{1, 2, \cdots, T\}, \tag{1}$$

where $[\cdot, \cdot]$ indicates a concatenation of two feature vectors. Note that the initial memory tokens m^0 are trainable parameters learnt during training.

The first phase of IFC is Encode-Receive, which processes frames individually as follows:

$$[f_t^l, \widehat{m}_t^l] = \mathcal{E}^l([f_t^{l-1}, m_t^{l-1}]), \tag{2}$$

where $\mathcal{E}^l$ denotes the l-th Encode-Receive layer. With a self-attention computed over the frame pixel locations and the memory tokens, the information of each frame can be passed to the memory tokens and vise-versa.

The outputs of Encode-Receive are grouped by memory indices and formulate the inputs for Gather-Communicate layer. The grouping can be understood as a decomposition of memory tokens, and becomes computationally beneficial when the total size of gathered memory tokens increases.

$$[m_1^l(i), m_2^l(i), \cdots, m_T^l(i)] = \mathcal{G}^l([\widehat{m}_1^l(i), \widehat{m}_2^l(i), \cdots, \widehat{m}_T^l(i)]), \qquad i \in \{1, 2, \cdots, M\}, \tag{3}$$

where $\mathcal{G}^l$ denotes the l-th Gather-Communicate layer. The processed outputs are redistributed by the originated frame and get concatenated as $m_t = [m_t(1), m_t(2), \cdots, m_t(M)]$. Unlike Encode-Receive, Gather-Communicate utilizes the attention mechanism to convey the information from different frames over the input clip.

Defining a the l-th inter-frame encoder block (IFC^l) as $\mathcal{E}^l$ followed by $\mathcal{G}^l$, the stack of N_E encoder blocks can be inductively formulated as:

$$[f_t^l, m_t^l] = \text{IFC}^l([f_t^{l-1}, m_t^{l-1}]), \qquad 1 \leq l \leq N_E, \tag{4}$$

where $[f_t^{N_E}, m_t^{N_E}]$ is the final result. The stacking of multiple encoders brings communications between frames, thus each frame can have coincidence to the other, specifying the identities of instances in a given clip.

Complexity comparison. In Table 1, we analyze the computational complexity of transformer encoder variants applied for video input in terms of the Big-O complexity and FLOPs. The complexity of the original transformer encoder layer [14] is $\mathcal{O}(C^2N + CN^2)$, where N is the number of inputs. Without any communication between frames, *No Comm*, it shows the smallest amount of computation ($\mathcal{O}(C^2THW + CT(HW)^2)$). As indicated as *Full THW* in Table 1, the complexity of VisTR [11] that performs a full space-time self-attention is $\mathcal{O}(C^2(THW) + C(THW)^2)$ thus either a higher resolution or an increase of number of input frames leads to a massive increase in computations. VisTR bypasses the problem by highly reducing the input resolution and utilizing GPUs with tremendous

[1]Measured using `flop_count` function of `fvcore==0.1.5`.

memory capacity. However, as such solutions cannot resolve the fundamental issues, it is impractical to real-world videos. Moreover, VisTR remains as a complete offline strategy because it takes the entire video as an input.

An intriguing improvement for the naïve full self-attention would be the decomposition of the attention into space and time axis [24, 16]. In *Decompose T-HW*, we decompose attention computation into spatial and temporal attention. The complexity of the separation of space-time leads to the sum of the two transformer encoder: $\mathcal{O}(T(C^2(HW)+C(HW)^2))$ and $\mathcal{O}(HW(C^2T+CT^2))$. In comparison to the full self-attention, the decomposition lowers the computational growth relative to the number of frames.

Our encoder, IFC, that communicates between frames using the memory tokens leads to a huge benefit to the total computations adding only a small amount of computation over *No Comm* while providing sufficient channels for communication. The complexity of each phase in our proposed encoder is: $\mathcal{O}(C^2T(HW+M)+CT(HW+M)^2)$ for Encode-Receive and $\mathcal{O}(C^2TM+CT^2M)$ for Gather-Communicate respectively. Assuming that M is kept small (e.g., 8), the computation needed for Gather-Communicate can be neglected, while the complexity of Encode-Receive can be approximated to $\mathcal{O}(C^2THW+CT(HW)^2)$ as shown in Table 1. Finally, with respect to the number of frames of the input, we can expect approximate linear increase rather than the high increase of computation occurred in VisTR.

Decoders and output heads As depicted in Fig. 1, the transformer decoder of our model is stacked with N_D layers [14]. Contrary to VisTR, where the number of object queries increases proportionally to the number of frames, our model receives learnt encodings of fixed size N_q for *object queries*. Also, by utilizing these encodings throughout the entire frames, our model can effectively deal with clips of various lengths. A set of projection matrices are applied to $\{f_t^{N_E}, m_t^{N_E}\}_{t=1}^T$ for the generation of keys and values. The object queries turn into *output embeddings* by the transformer decoder, and the embeddings are eventually used as an input to the output heads.

There are two output heads on top of the transformer decoder, a class head and a segmentation head, each composed of two fully-connected layers. The output embeddings from the transformer decoder are independently inserted to the heads, resulting in N_q predictions per a clip. The class head outputs a class probability distribution of instances $\hat{p}(c) \in \mathbb{R}^{N_q \times |\mathbb{C}|}$. Note that the possible classes $\mathbb{C} \ni c$ include *no object* $\varnothing$ class in addition to the given classes of a dataset.

The segmentation head generates N_q conditional convolutional weights $w \in \mathbb{R}^{N_q \times C}$ in a manner similar to [19, 7]. For the conditional convolution, the output feature of the encoder $\{f_t^{N_E}\}_{t=1}^T$ is reused by undoing the flatten operation. For the upsampling, the encoder feature passes through fpn-style [25] *spatial decoder* without temporal connections resulting in T feature maps that are 1/8 of the input resolution. Finally, the resulting feature maps are convolved with each convolutional weights to generate segmentation mask as follows:

$$\hat{s_i} = \{f_t' \circ w_i\}_{t=1}^T, \tag{5}$$

where w_i is i-th convolutional weight, $\circ$ indicate 1×1 spatial convolution operation, and the result $\hat{s_i}$ is a spatial-temporal object mask in shape of $\mathbb{R}^{T \times H' \times W'}$ where $H' = \frac{H_0}{8}$, $W' = \frac{W_0}{8}$. Note that, for an instance, a common weight is applied throughout the video clip. Our spatial decoder is of instance-agnostic design which gets highly efficient than decoders of instance-specific designs [13, 11, 12, 10] as the number of detected instances increases. Meanwhile, thanks to our segmentation head which specifies and captures the characteristics of an instance, IFC can conduct both segmentation and tracking at once within a clip.

3.2 Instance matching and loss

To train our network, we first assign the ground truth for each instance estimation and then a set of loss function between each the ground truth and prediction pair. For a given input clip, our model generate a fixed-size set of class-labeled masks $\{\hat{y}_i\}_{i=1}^{N_q} = \{(\hat{p}_i(c), \hat{s}_i)\}_{i=1}^{N_q}$. The ground truth set of the clip can be represented as $y_i = (c_i, s_i)$; c_i is the target class label including $\varnothing$, and s_i is the target mask which is down-sampled to the size of the prediction masks for efficient similarity calculation. One-to-one bipartite matching between the prediction set $\{\hat{y}_i\}_{i=1}^{N_q}$ and the ground truth set $\{y_i\}_{i=1}^K$ is performed to find the best assignment of a prediction to a ground truth. The objective can be formally

described as:

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_{N_q}}{\arg\max} \sum_{i=1}^{K} \mathrm{sim}(y_i, \hat{y}_{\sigma(i)}), \tag{6}$$

where $\mathrm{sim}(y_i, \hat{y}_{\sigma(i)})$ refers a pair-wise similarity over a permutation of $\sigma \in \mathfrak{S}_{N_q}$. Following prior work [26, 13, 19], the bipartite matching is efficiently computed using Hungarian algorithm [18]. We find that box-based similarity measurement as used in DETR [13] shows weaknesses in matching instances in video clip due to the case of occlusion and disappear-and-reappear. Therefore, we define $\mathrm{sim}(y_i, \hat{y}_{\sigma(i)})$ to be mask-based term as $\mathbb{1}_{\{c_i \neq \varnothing\}}[\hat{p}_{\sigma(i)}(c_i) + \lambda_0 \textsc{Dice}(s_i, \hat{s}_{\sigma(i)})]$, where $\textsc{Dice}$ denotes dice coefficients [27].

Given the optimal assignment $\hat{\sigma}$, we refer to the K matched predictions and $(N_q - K)$ non-matched predictions as positive and negative pairs respectively. The positive pairs aim to predict the ground truth masks and classes while the negative pairs are optimized to predict the $\varnothing$ class. The final loss is a sum of the losses from positive pairs and negative pairs where each can be computed as follows:

$$\begin{aligned} \mathcal{L}_{pos} &= \sum_{i=1}^{K} [\underbrace{-\log \hat{p}_{\hat{\sigma}(i)}(c_i)}_{\text{Cross-entropy loss}} + \lambda_1 \underbrace{(1 - \textsc{Dice}(s_i, \hat{s}_{\hat{\sigma}(i)}))}_{\text{Dice loss [27]}} + \lambda_2 \underbrace{\textsc{Focal}(s_i, \hat{s}_{\hat{\sigma}(i)})}_{\text{Sigmoid-focal loss [28]}}], \\ \mathcal{L}_{neg} &= \sum_{i=k+1}^{N_q} [-\log \hat{p}_{\hat{\sigma}(i)}(\varnothing)]. \end{aligned} \tag{7}$$

As $(N_q - K)$ is likely to be much greater than K, we down-weight $\mathcal{L}_{neg}$ by a factor of 10 to resolve the imbalance, following prior work [13]. The goal of video instance segmentation [1] is to maximize the space-time IoU between a prediction and a ground truth mask. Therefore, our mask-related losses (Dice loss and Sigmoid-focal loss) are spatio-temporally calculated over an entire clip, rather than averaging the losses that are accumulated frame-by-frame.

3.3 Clip-level instance tracking

To infer a video input that is longer than the clip length, we match instances using the predicted masks of overlapping frames. Let $\mathcal{Y}_I$ and $\mathcal{Y}_A$ be the result sets of clip I and A excluding the $\varnothing$ class. The goal is to perform matching of same identities between pre-collected instance set $\mathcal{Y}_I$ and $\mathcal{Y}_A$. We first calculate the matching scores which are space-time soft IoU at intersecting frames between $\mathcal{Y}_I$ and $\mathcal{Y}_A$. Then, we find optimal paired indices $\hat{\sigma}_{\mathcal{S}}$ using Hungarian algorithm [18] to the gathered matching score $\mathcal{S} \in [0,1]^{|\mathcal{Y}_I| \times |\mathcal{Y}_A|}$. We update $\mathcal{Y}_I(i)$ by concatenating $\mathcal{Y}_A(\hat{\sigma}_{\mathcal{S}}(i))$ if $\mathcal{S}(i, \hat{\sigma}_{\mathcal{S}}(i))$ is above a certain threshold, and add non-matched prediction sets to $\mathcal{Y}_I$ as *new* instances. Note that a previous per-clip model (MaskProp [10]) also utilizes soft IoU for tracking instances, but the matching scores are computed per-frame and averaged for intersecting frames. Different from MaskProp, using space-time soft IoU leads to an accurate tracking as it can better represent the definition of mask similarities between clips which brings at most 2% AP increase. The overall tracking pipeline can be effectively implemented in GPU-friendly manner.

4 Experiments

In this section, we evaluate the proposed method using YouTube-VIS 2019 and 2021 [1]. We demonstrate the effectiveness of our model regarding both accuracy and speed. We further examine how different settings affect the overall performance and efficiency of IFC encoder. Unless specified, all models for measurements used $N_E = 3, N_D = 3$, stride of 1, and ResNet-50.

4.1 Main Results

YouTube-VIS 2019 evaluation results. We compare our proposed IFC to the state-of-the-art models in the video instance segmentation task on YouTube-VIS 2019 `val` in Table 2 (a). We measure the accuracy by AP and our model sets the highest score among all online, near-online, and offline models while presenting the fastest runtime. As mentioned earlier, IFC is highly efficient during the inference thanks to three advantages: (1) memory token-based decomposition for transformer encoder (2) instance-agnostic spatial decoder (3) GPU-friendly instance matching. Moreover, our

Table 2: Evaluations on various settings.

(a) AP and FPS on YouTube-VIS 2019 `val` set. For fairness, FPS is measured on a same machine, using a single RTX 2080Ti GPU. We used the official codes and checkpoints provided by the authors for the measurements. We report the clip settings of [11, 10]. T: window size.

Method (Settings)		Backbone [29]	FPS[2]	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
MaskTrack R-CNN [1]		ResNet-50	26.1	30.3	51.1	32.6	31.0	35.5
MaskTrack R-CNN [1]		ResNet-101	-	31.8	53.0	33.6	33.2	37.6
SipMask [2]		ResNet-50	35.5	33.7	54.1	35.8	35.4	40.1
SG-Net [4]		ResNet-50	-	34.8	56.1	36.8	35.8	40.8
SG-Net [4]		ResNet-101	-	36.3	57.1	39.6	35.9	43.0
CrossVIS [3]		ResNet-50	-	36.3	56.8	38.9	35.6	40.7
CrossVIS [3]		ResNet-101	-	36.6	57.3	39.7	36.0	42.0
STEm-Seg [30]		ResNet-101	3.0	34.6	55.8	37.9	34.4	41.6
CompFeat [31]		ResNet-50	-	35.3	56.0	38.6	33.1	40.3
VisTR [11]	(T=36)	ResNet-50	51.1	36.2	59.8	36.9	37.2	42.4
VisTR [11]	(T=36)	ResNet-101	43.5	40.1	64.0	45.0	38.3	44.9
MaskProp [10]	(T=13)	ResNet-50	-	40.0	-	42.9	-	-
MaskProp [10]	(T=13)	ResNet-101	-	42.5	-	45.6	-	-
Ours$_{\text{near-online}}$	(T=5)	ResNet-50	46.5	41.0	62.1	45.4	43.5	52.7
Ours$_{\text{offline}}$	(T=36)	ResNet-50	107.1	42.8	65.8	46.8	43.8	51.2
Ours$_{\text{offline}}$	(T=36)	ResNet-101	89.4	44.6	69.2	49.5	44.0	52.1

(b) Accuracy on YTVIS 2021 `val` set

	AP	AP_{50}	AP_{75}
MaskTrack-RCNN	28.6	48.9	29.6
SipMask	31.7	52.5	34.0
CrossVIS	34.2	54.4	37.9
Ours	36.6	57.9	39.3

(c) Bipartite matching

	AP
Box-based	37.2
Mask-based	39.4

(d) Effect of strides

		AP	AP_{75}	FPS
T = 5	S = 3	40.9	45.0	72.7
T = 10	S = 5	41.1	44.5	83.0
T = 15	S = 8	42.0	45.9	92.5
T = 20	S = 10	42.4	46.9	95.7

model does not make use of any heavy modules such as deformable convolutions [32] or cascading networks [33]. Thanks to these advantages, IFC achieves an outstanding runtime, which is faster speed than online models [1, 2].

During the inference, our method is able to freely adjust the length of the clip (T) as needed. If the input clip length is set to contain entire video frames, our method becomes an offline method (like VisTR [11]) that processes the entire video in one shot. As the offline inference can skip matching between clips and maximize the GPU utilization, our method represents surprisingly fast runtime (107.1 FPS). On the other hand, if the application requires instant outputs given a video stream, we can reduce the clip length to make our method near-online. In the near-online scenario with $T = 5$, our system is still able to process a video in real-time (46.5 FPS) with only a small delay.

YouTube-VIS 2021 evaluation results. The recently introduced dataset YouTube-VIS 2021 is an improved version of YouTube-VIS 2019. The newly added videos in the dataset include higher number of instances and frames. In Table 2 (b), we refer the results reported in [3], which evaluated [1, 2] using official implementations. Again, our model achieves the best performance.

4.2 Ablation Study

In this section, we provide ablation studies and discuss how different settings impact the overall performance. The experiments are conducted using YouTube-VIS 2019 `val` set. For every ablation studies, we report the mean of five runs as the results may vary by each run due to the insufficient number of training and testing set of YouTube-VIS dataset.

Box-based and mask-based bipartite matching. We observe how the different policies for bipartite matching affect the performance. As our model does is a box-free method, we adjust our model

[2] We follow `detectron2` [34] for measuring FPS.

Table 3: Encoder variations. We show how different encoders affect the overall performance.

(a) Various encoders taking clips of different lengths (see Table 1)

	T=5			T=10			T=15			T=20		
	AP	AP_{75}	FPS	AP	AP_{75}	FPS	AP	AP_{75}	FPS	AP	AP_{75}	FPS
No Comm	37.4	39.9	38.1	38.8	41.6	40.8	39.3	41.7	46.7	39.6	41.9	52.9
Full THW	37.2	40.0	37.6	38.8	41.2	35.5	39.8	42.6	32.9	39.7	42.8	34.8
Decomp T-HW	37.2	39.8	35.7	38.3	40.9	37.9	38.5	41.5	42.6	39.0	41.9	49.4
IFC	39.0	42.7	36.3	39.6	43.0	38.9	39.8	43.0	43.7	40.4	43.4	50.2

(b) Image instance segmentation on COCO `val` set

	AP^{coco}	AP^{coco}_{50}
w/o mem	35.0	56.6
w/ mem	35.1	56.5

(c) Number of memory tokens (AP)

	T=5	T=10	T=15	T=20
M=1	37.6	39.2	39.4	39.4
M=2	37.9	39.2	39.6	39.8
M=4	38.0	39.5	39.7	39.9
M=8	**39.0**	**39.6**	**39.8**	**40.4**
M=16	38.1	39.1	39.7	39.9

(d) Index-wise memory decomposition

	T=5	T=10	T=15	T=20
Unified	38.1	38.9	39.7	39.9
Decomp	**39.0**	**39.6**	**39.8**	**40.4**

to predict bounding boxes similar to VisTR [11] and conduct bipartite matching [18, 13] using the predicted boxes. The change of optimization from mask-based to box-based brings a noticeable performance drop as shown in Table 2 (c). With the VIS-centric design, the mask-based optimization shows more robustness than box-based optimizations under typical video circumstances such as instances with heavy overlaps and partial occlusions.

Differing window strides. In addition to the clip length (T), we further optimize our runtime placing a stride S between clips, as shown in Table 2 (d). IFC can be used in a near-online manner, which takes clips that are consecutively extracted from a video. The placement of a larger stride reduces temporal intersections, which lessens computational overheads but also causes difficulty in matching instances. By enlarging the stride from $S = 1$ to $S = 3$, IFC accomplishes approximately 150% speed improvement with only 0.1% AP drop. The tendency of high speed gain and low accuracy drop persists under various conditions. Therefore, our model can be applied to conditions where the enlargement of strides is necessary, *i.e.*, using devices that are not powerful enough but has to maintain high inference speed.

Various decomposition strategies of encoders. In Table 1, we observed the computational gaps derived from the decomposition of the encoder layers. Extending Table 1, we now investigate the how the decomposition strategies affect the accuracy in Table 3.

The models are evaluated with variety of window sizes ($T = 5, 10, 15, 20$) as an increase of window size T has pros and cons. When matching predictions from different clips, greater T is advantageous due to an enlargement of temporal intersections between clips. On the contrary, frames in longer clips are likely to be composed of diverse appearances, which disrupt tracking and segmenting instances within a clip. Therefore, the key to the performance enhancement is to cope with the appearance changes by precisely encoding and correlating space-time inputs.

As shown in Table 3 (a), the *full* self-attention [11] surpasses the encoder without communications as the length of clips increase. However, the enlargement of the window size highly slows down the inference speed, and the improvements are marginal that the tremendous computation and memory usage cannot be compensated. The *decomposition of space-time* maintains comparable speed even if the window is large, but fails to achieve high accuracy.

Our model is shows fast inference as the only additional computations of IFC are from utilizing a small number of memory tokens. Furthermore, by effectively encoding the space-time inputs with the communications between frames, IFC can take advantages of enlarging the window size, and surpasses other encoders.

Memory tokens. We also study the effects of utilizing memory tokens. As mentioned, the motivation of using the memory tokens is to build communications between frames. Different from the video instance segmentation task, the image segmentation task is consisted of a single frame. Therefore, the use of the memory tokens does not lead to improvements to the image instance segmentation

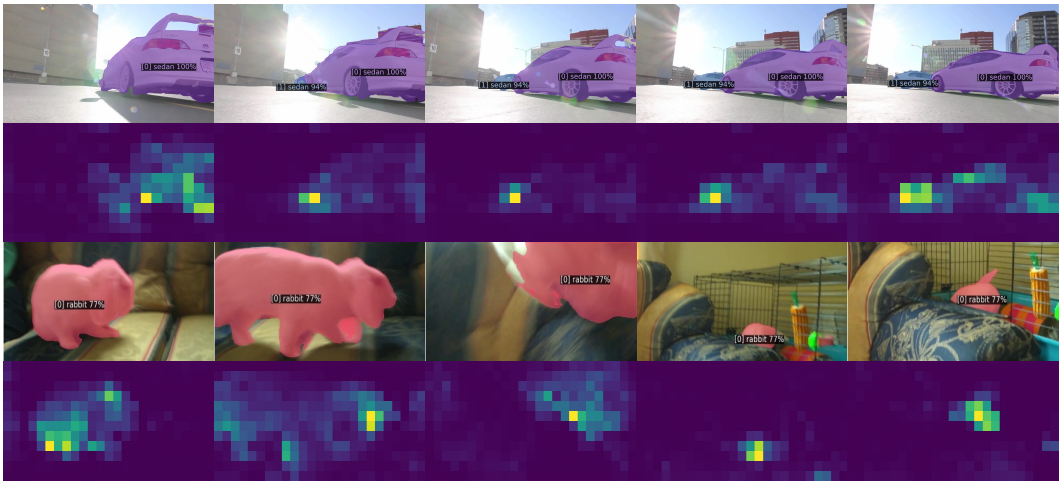

Figure 2: Visualizations of results and attention maps of memory tokens.

task as mutual communications cannot be solely made (see Table 3 (b)). Meanwhile, the utilization of the memory tokens achieves great improvements by effectively passing the information between frames. Results in Table 3 (a, c) demonstrate that the use of memory tokens achieves higher accuracy than the encoder without any communications (*No comm*), which emphasizes the importance of the communications. We evaluate how the size of the memory tokens affect the overall accuracy in Table 3 (c) and set the default size of the tokens M to be 8.

In Section 3.1, we demonstrated the formulation of the inputs for Gather-Communicate layer, which groups the outputs of Encode-Receive by memory indices. As aforementioned, the formulation can be considered as a decomposition of memory tokens: insertion to the Gather-Communicate layer by separate M groups each consisting of T tokens. In Table 3 (d), we investigate the impact of inserting the unified MT tokens as a whole. Compared to the *unified* insertion, the *decomposition* brings better accuracy as the memories of same indices have more correspondences, which ease the encoders to build attentions in between.

We choose a memory index attending foreground instances and visualize the attention map in Fig. 2. As shown in the results of the upper clip, we find that the memory token has more interests to instances that are relatively difficult to detect; it more attends the heavily occluded car at the rear. The clip at the bottom is composed of frames with huge motion blurs and appearance changes. With the communications of memory tokens, IFC successfully tracks and segments the rabbit.

5 Conclusion

In this paper, we have proposed a novel video instance segmentation network using Inter-frame Communication Transformers (IFC), which alleviates full space-time attention and successfully builds communications between frames. Finally, our network presents a rapid inference and sets the new state-of-the-art on the YouTube-VIS dataset. For the future work, we plan to integrate temporal information, which indeed would take a step further to the human video understanding.

Broader Impact

Our framework is designed for the VIS task, which targets to classify and segment foreground instances of predefined classes. Recently, while investigating the capabilities of transformers, many disregard the importance of efficiency and take inputs of tremendous sizes. In comparison, IFC focuses on reducing the overall computation while improving the performance. We believe our network can positively impact many industrial fields that require high accuracy and speed, *i.e.*, alert system, autonomous driving, robotics. We want to note that for the community to move in the right direction, the studies on VIS should be aware of potential misuses which violates personal privacy.

COCO [35], YouTube-VIS [1], `detectron2` **[34] license:** CC-4.0, CC-4.0, Apache-2.0

References

[1] Yang, L., Y. Fan, N. Xu. Video instance segmentation. In *ICCV*. 2019.

[2] Cao, J., R. M. Anwer, H. Cholakkal, et al. Sipmask: Spatial information preservation for fast image and video instance segmentation. In *ECCV*. 2020.

[3] Yang, S., Y. Fang, X. Wang, et al. Crossover learning for fast online video instance segmentation. *arXiv preprint arXiv:2104.05970*, 2021.

[4] Liu, D., Y. Cui, W. Tan, et al. Sg-net: Spatial granularity network for one-stage video instance segmentation. In *CVPR*. 2021.

[5] He, K., G. Gkioxari, P. Dollar, et al. Mask r-cnn. In *ICCV*. 2017.

[6] Bolya, D., C. Zhou, F. Xiao, et al. Yolact: Real-time instance segmentation. In *ICCV*. 2019.

[7] Tian, Z., C. Shen, H. Chen. Conditional convolutions for instance segmentation. In *ECCV*. 2020.

[8] Chen, H., K. Sun, Z. Tian, et al. Blendmask: Top-down meets bottom-up for instance segmentation. In *CVPR*. 2020.

[9] Luo, W., J. Xing, A. Milan, et al. Multiple object tracking: A literature review. *Artificial Intelligence*, 2020.

[10] Bertasius, G., L. Torresani. Classifying, segmenting, and tracking object instances in video with mask propagation. In *CVPR*. 2020.

[11] Wang, Y., Z. Xu, X. Wang, et al. End-to-end video instance segmentation with transformers. In *CVPR*. 2020.

[12] Lin, H., R. Wu, S. Liu, et al. Video instance segmentation with a propose-reduce paradigm. *arXiv preprint arXiv:2103.13746*, 2021.

[13] Carion, N., F. Massa, G. Synnaeve, et al. End-to-end object detection with transformers. In *ECCV*. 2020.

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

[15] Wang, H., Y. Zhu, B. Green, et al. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*. 2020.

[16] Bertasius, G., H. Wang, L. Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021.

[17] Wang, X., R. Girshick, A. Gupta, et al. Non-local neural networks. In *CVPR*. 2018.

[18] Kuhn, H. W. The hungarian method for the assignment problem. In *Naval research logistics quarterly*. 1955.

[19] Wang, H., Y. Zhu, H. Adam, et al. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *CVPR*. 2021.

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

[21] Ranftl, R., A. Bochkovskiy, V. Koltun. Vision transformers for dense prediction. *arXiv preprint arXiv:2103.13413*, 2021.

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

[23] Kirillov, A., K. He, R. Girshick, et al. Panoptic segmentation. In *CVPR*. 2019.

[24] Tran, D., H. Wang, L. Torresani, et al. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*. 2018.

[25] Lin, T.-Y., P. Dollar, R. Girshick, et al. Feature pyramid networks for object detection. In *CVPR*. 2017.

[26] Stewart, R., M. Andriluka, A. Y. Ng. End-to-end people detection in crowded scenes. In *CVPR*. 2016.

[27] Milletari, F., N. Navab, S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*. 2016.

[28] Lin, T.-Y., P. Goyal, R. Girshick, et al. Focal loss for dense object detection. In *ICCV*. 2017.

[29] He, K., X. Zhang, S. Ren, et al. Deep residual learning for image recognition. In *CVPR*. 2016.

[30] Athar, A., S. Mahadevan, A. Ošep, et al. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *ECCV*. 2020.

[31] Fu, Y., L. Yang, D. Liu, et al. Compfeat: Comprehensive feature aggregation for video instance segmentation. *arXiv preprint arXiv:2012.03400*, 2020.

[32] Dai, J., H. Qi, Y. Xiong, et al. Deformable convolutional networks. In *ICCV*. 2017.

[33] Cai, Z., N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*. 2018.

[34] Wu, Y., A. Kirillov, F. Massa, et al. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[35] Lin, T.-Y., M. Maire, S. Belongie, et al. Microsoft coco: Common objects in context. In *ECCV*. 2014.

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]
   (b) Did you describe the limitations of your work? [Yes] Sec. 5 contains our future work.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We included potential misuses in Sec. 5.
   (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? [Yes] We stated the assumptions and motivations of our work in Sec. 1 and Sec. 3.1.
   (b) Did you include complete proofs of all theoretical results? [Yes] In addition to the Big-O complexity in Sec. 3.1, we included actual results in Sec. 4.
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] We listed the data and instructions in Sec. 4 and Appendix, and the code will be released upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specified details in Sec. 4 and Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In Appendix, we reported standard deviations of the results in Table 3 (a, c, d).
   (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] We included the type of resources used for training and inference in Appendix and Table 2.
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 cited the papers and repositories that are used.
   (b) Did you mention the license of the assets? [Yes] We mentioned the licenses at the end.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] New assets are not included.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]

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] Not the scope of this paper.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] Not the scope of this paper.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] Not the scope of this paper.