

Temporal-aware Query Routing for Real-time Video Instance Segmentation

Zesen Cheng^{1,3} Kehan Li⁴ Yian Zhao^{1,3} Hang Zhang⁴ Chang Liu⁵ Jie Chen^{1,2,3} ✉

¹ School of Electronic and Computer Engineering, Peking University, Shenzhen, China ² Pengcheng Laboratory, Shenzhen, China

³ AI for Science (AI4S)-Preferred Program, Peking University Shenzhen Graduate School, China

⁴ Alibaba Group, Hangzhou, China ⁵ Department of Automation and BNRist, Tsinghua University, Beijing, China

cyanlaser@stu.pku.edu.cn jiechen2019@pku.edu.cn

Abstract

With the rise of applications such as embodied intelligence, developing high real-time online video instance segmentation (VIS) has become increasingly important. However, through time profiling of the components in advanced online VIS architecture (i.e., transformer-based architecture), we find that the transformer decoder significantly hampers the inference speed. Further analysis of the similarities between the outputs from adjacent frames at each transformer decoder layer reveals significant redundant computations within the transformer decoder. To address this issue, we introduce **Temporal-Aware query Routing (TAR)** mechanism. We embed it before each transformer decoder layer. By fusing the optimal queries from the previous frame, the queries output by the preceding decoder layer, and their differential information, TAR predicts a binary classification score and then uses an argmax operation to determine whether the current layer should be skipped. Experimental results demonstrate that integrating TAR into the baselines achieves significant efficiency gains (24.7 → 34.6 FPS for MinVIS, 22.4 → 32.8 FPS for DVIS++) while also improving performance (e.g., on YoutubeVIS 2019, 47.4 → 48.4 AP for MinVIS, 55.5 → 55.7 AP for DVIS++). Furthermore, our analysis of the TAR mechanism shows that the number of skipped layers increases as the differences between adjacent video frames decrease, which suggests that our method effectively utilizes inter-frame differences to reduce redundant computations in the transformer decoder.

1. Introduction

Video Instance Segmentation (VIS) [45] is an advanced computer vision technology that aims to provide precise segmentation masks for each object instance in a video and to associate the same object across frames. This technology is widely utilized in fields such as medical surgery, em-

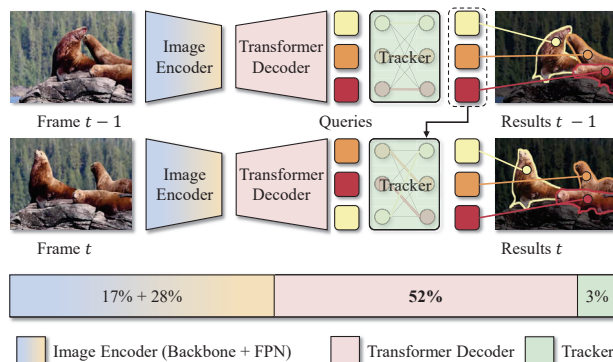


Figure 1. Time profiling of components in the advanced online VIS method MinVIS shows: Image Encoder (Backbone + FPN) 45% (17%+28%), Transformer Decoder 52%, Tracker 3%. This distribution identifies the transformer decoder as the key computational bottleneck, whose optimization can greatly boost efficiency.

bodied intelligence, and human-computer interaction. For instance, in the context of embodied intelligence, VIS can precisely locate various objects in real time within a scene, allowing agents to achieve better environmental understanding and perform corresponding actions based on the accurate positions of those objects. As these applications are further adopted, fast and accurate low-latency object localization becomes increasingly crucial to ensure the system’s safety and responsiveness while improving user experience.

In general, VIS has three paradigms: offline, semi-online/near-online, and online. Offline VIS [11, 17] methods require complete video data for segmentation, which clearly does not meet the demands for real-time response. Semi-online/near-online VIS methods [18, 31] focus on clip-level object segmentation and tracking. However, they fail to satisfy the stringent real-time requirements of streaming scenarios due to significant delays in segment acquisition. For example, in a 10 FPS video stream, obtaining 5 frames takes 0.5 seconds, resulting in a minimum reaction time of 0.5 seconds for semi-online methods operating at this granularity, which is unacceptable for real-time appli-

✉ Corresponding Author

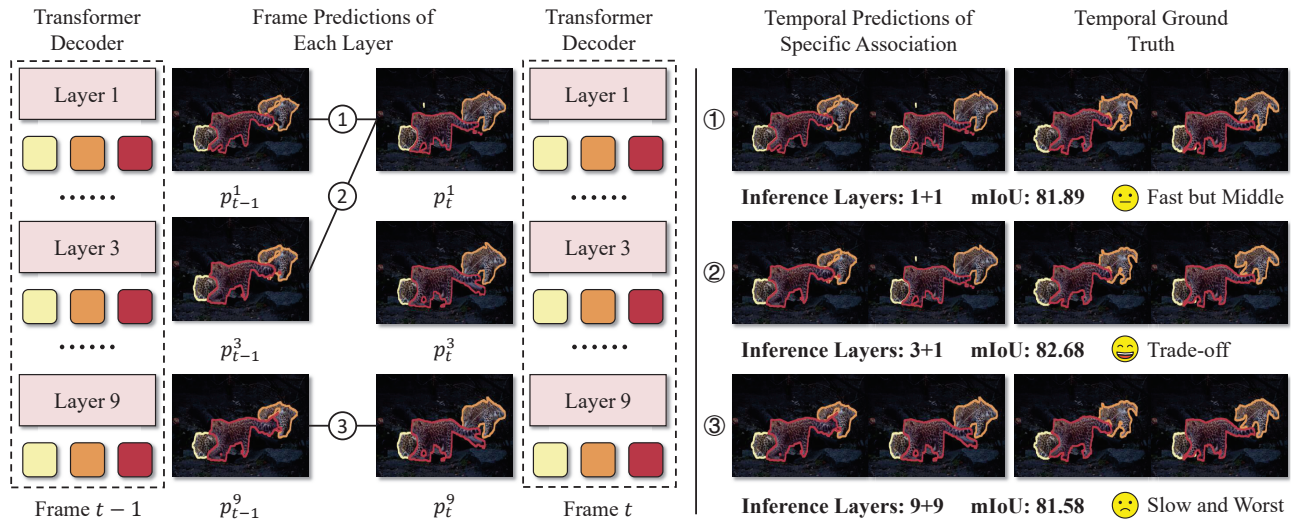


Figure 2. **Carefully selecting transformer decoder layer combinations across adjacent frames can optimize efficiency while maintaining performance.** We evaluate three different transformer decoder layer combinations between adjacent frames: “1+1”, “3+1”, and “9+9”. The results show that the combination with the highest inference cost, “9+9” (81.58 mIoU), does not yield the best performance, while the combination with moderate inference cost, “3+1” (82.68 mIoU), achieves the best results.

cations. Online VIS methods operate at the frame level for segmentation and tracking, providing timely feedback for each video frame. This capability meets the high real-time requirements of dynamic scenarios effectively.

However, time-consuming analysis of components in the advanced online VIS method MinVIS (a simple Transformer-based architecture) shows: Image Encoder (Backbone + FPN) 45% (17% + 28%), Transformer Decoder 52%, Tracker 3% (Figure 1). This identifies the image encoder and transformer decoder as computational bottlenecks. Given the encoder backbone’s need for large-scale pre-training and the FPN’s complex design, we focus on optimizing the transformer decoder.

We observe that the transformer decoder is composed of stacked transformer decoder layers, where each layer is capable of outputting a set of queries for tracking, segmentation and classification tasks. This naturally raises a question: Are all layers necessary, or is there redundant computation? To explore this, we evaluate three combinations of outputs from different transformer decoder layers across adjacent frames: “1+1”, “3+1”, and “9+9” (where “m+n” represents the output from the m-th layer of the previous frame and the n-th layer of the current frame) and assess their performance by matching these combinations with the Ground Truth and computing the IoU. In Figure 2, we observe that the “3+1” combination achieves the optimal balance between performance and inference speed, demonstrating that *strategically selecting transformer decoder layer combinations across adjacent frames can enhance efficiency without compromising performance.*

Next, we have to consider another question: What cri-

terion should guide the selection of transformer decoder layers? The transformer decoder generates queries for getting segmentation and classification results, with each query essentially sampling specific regions (e.g., objects) in the image features. Since changes between adjacent video frames are typically minimal, the queries across these frames should also be highly similar. Assuming we have the optimal query from the previous frame, we can compare it with the queries generated at each layer. If a query from a particular layer shows high similarity to the current frame’s query, we can deem this layer’s query optimal and bypass computations in subsequent layers to eliminate redundancy.

To achieve this, we propose **Temporal-Aware query Routing (TAR)** and insert it before each transformer decoder layer to assess whether the layer is redundant and can be skipped. Specifically, TAR fuses the query from the previous frame, the query from the current frame, and the differential information between the two queries to predict a binary score. The argmax operation then determines whether to skip the layer. During training, we use the Straight-Through Gumbel-Softmax trick to ensure the differentiability of the argmax operation. Through experiments integrating TAR into two baseline models (MinVIS and DVIS++), we find that it not only significantly enhances overall inference efficiency (from 24.7 to 34.6 FPS for MinVIS and from 22.4 to 34.8 FPS for DVIS++) but also achieves comparable or even slightly better performance across multiple datasets. For example, TAR improves MinVIS/DVIS++ by 1.0/0.2 AP on YouTubeVIS 2019, 0.1/0.8 AP on YouTubeVIS 2021, and 2.4/0.9 AP on OVIS. Additionally, through experiments in Figure 5, we observed that

the smaller the changes between frames, the fewer layers TAR skips. This indicates that our method effectively leverages inter-frame differences to eliminate redundant layers.

2. Related Works

2.1. Video Instance Segmentation

Video instance segmentation (VIS) is a comprehensive task that requires simultaneous segmentation, tracking, and classification of all object instances across video frames [45]. Early approaches to VIS can be broadly categorized into one-stage (e.g., [2, 7, 27]) and two-stage methods (e.g., [6, 45]). However, with the advent of transformers [8, 14], VIS has increasingly adopted attention-based architectures due to their ability to model global dependencies and provide elegant query-centric instance representations. For example, VisTR [41] extends DETR [8] to VIS by introducing an end-to-end framework that tracks objects across videos using instance queries. Transformer-based VIS methods have since evolved into three main paradigms: offline, semi-online, and online approaches.

Offline methods [10, 22, 47] process entire videos as spatiotemporal volumes, leveraging temporal coherence for improved accuracy. For instance, IFC [21] introduces memory tokens to propagate instance information across frames, while Seqformer [42] decomposes frame queries to enhance temporal modeling. Although offline methods achieve state-of-the-art performance, they are computationally expensive and memory-intensive, making them less practical for long videos. **Semi-online methods** [18, 31] segment video clips and perform tracking at the clip level, improving inference efficiency and performance in offline long video scenarios. However, they struggle in online settings due to the latency incurred while acquiring clips. For example, in a 10fps video stream, obtaining a 5-frame clip requires 0.5 seconds, making semi-online methods unsuitable for real-time applications. In contrast, **online methods** [19, 43, 51] process videos incrementally, either frame-by-frame or in short clips, focusing on efficient instance association across adjacent frames. For instance, MinVIS [19] employs the Hungarian algorithm [25] to match instance queries between consecutive frames. While slightly less accurate than offline methods, online approaches are significantly more efficient and scalable, making them suitable for real-world applications such as autonomous driving [49] and video editing [33].

In many practical applications, such as embodied intelligence and surgical robotics, it is often crucial for video instance segmentation models to respond promptly and efficiently. Therefore, we primarily focus on optimizing the online video instance segmentation paradigm. Online methods align better with real-time processing requirements because they can avoid the memory bottlenecks inherent in

offline approaches and the high latency associated with clip acquisition in semi-online methods.

2.2. Conditional Computation

Conditional Computation is a computational paradigm that dynamically activates a subset of model parameters or modules based on input data, rather than using the entire model for every computation [4, 5]. We primarily focus on early-exit technology, an implementation of conditional computation that places classifiers at intermediate layers of a model, enabling it to output results early when predictions reach sufficient confidence, thereby avoiding computations in subsequent layers [40, 44]. Early-exit can dynamically adjust computation paths based on input complexity: for simple inputs, the model exits early, skipping subsequent layers and significantly reducing computational load; for complex inputs, it utilizes more layers, achieving a balance between efficiency and accuracy [12, 15, 29, 35, 36]. This makes it highly advantageous for resource-constrained, real-time scenarios. Additionally, early-exit can be seamlessly integrated into existing deep neural networks by simply adding classifiers to intermediate layers, requiring no extensive architectural modifications, thus offering high practical value.

Based on our observation in Figure 2 that the transformer decoder exhibits some redundant layers, we specifically explore in this paper how to adopt advanced early-exit techniques, particularly SkipLayer-based methods [1, 26, 50], to bypass these redundant layers in the transformer decoder for VIS. This aims to enhance the inference efficiency of VIS methods, enabling these VIS models to be more effectively deployed in real-time practical applications.

3. Methods

In this section, we first introduce the overall pipeline of our online VIS in Section 3.1. Then, in Section 3.2, we describe how TAR is integrated into the existing online VIS framework, followed by detailed implementation specifics, such as differentiable operations during training and how redundant layers are skipped during inference.

3.1. Overall Pipeline

We primarily select representative online VIS models, MinVIS [19] and DVIS++ [52], as baselines to verify the effectiveness of our TAR. The architecture of two baselines consists of an image encoder, a transformer decoder, and a tracker, with our method primarily targeting the transformer decoder. To provide a concrete and vivid description of the classical architecture and how TAR enhances inference efficiency, we illustrate the overall pipeline in Figure 3.

Image Encoder. The image encoder primarily consists of a backbone and a pixel decoder. In an online scenario, assuming we have captured the t -th frame of a video stream, this frame is fed into the backbone to extract features: $\{\mathcal{F}_t^{b_2} \in$

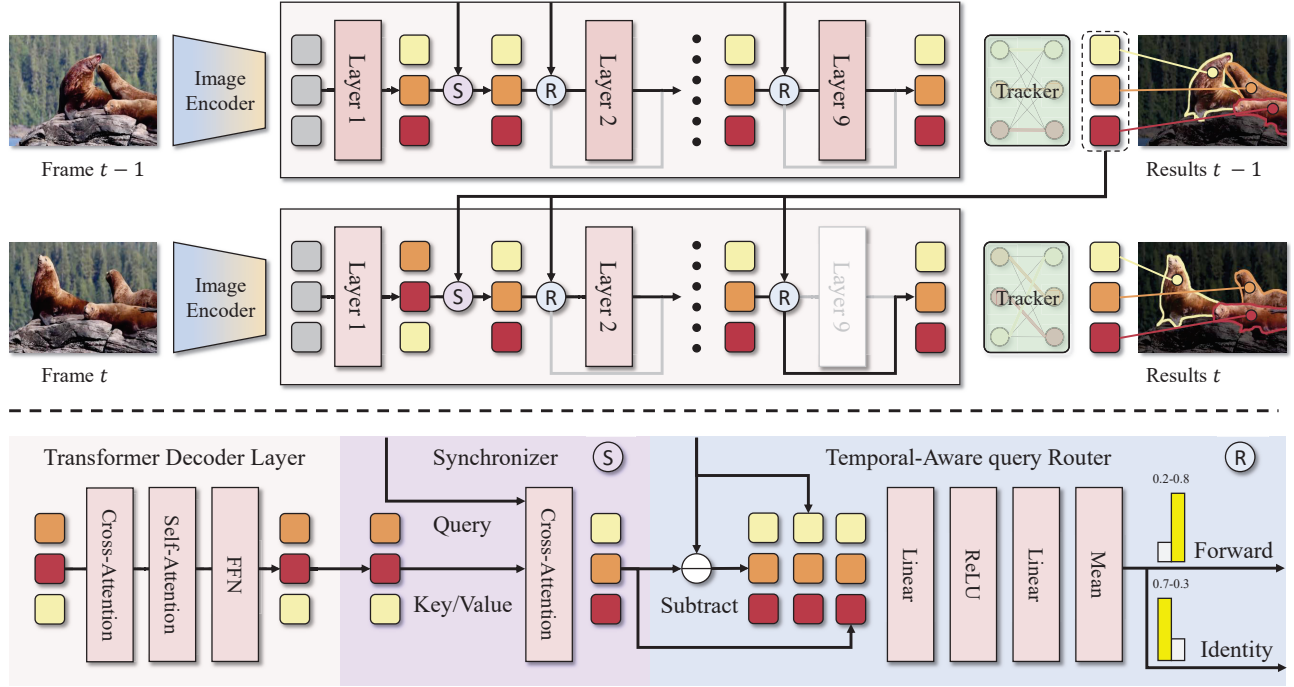


Figure 3. **Overall pipeline of online VIS with TAR.** We illustrate the architecture of online VIS and show how we integrate TAR to enhance overall efficiency. TAR is introduced before each transformer decoder layer for skipping redundant transformer decoder layers. To achieve this, the current frame’s queries are first synchronized with those queries from the previous frame to obtain inter-frame differential information. Subsequently, the synchronized queries of the current frame are fed into the TAR of each transformer decoder layer, and are concatenated with the previous queries along with the differential information between two group of queries. This combined information is then processed through a MLP and averaged to produce a binary score, which determines whether each layer should be skipped.

$\mathbb{R}^{\frac{H}{8} \times \frac{W}{8} \times d_2}, \mathcal{F}_t^{b_3} \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times d_3}, \mathcal{F}_t^{b_4} \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{32} \times d_4}$, where H , W , and d denote height, width, and feature dimension, respectively. Subsequently, a Feature Pyramid Network (FPN) progressively integrates backbone features to generate high-quality multi-scale pixel features. Various FPN variants are available, such as Vanilla FPN [28], Semantic FPN [24], FaPN [20], and BiFPN [37]. Here, following Mask2Former [11], both MinVIS and DVIS++ adopt Multi-scale Deformable Attention from Deformable DETR [54]:

$$(\mathcal{F}_t^{o_2}, \mathcal{F}_t^{o_3}, \mathcal{F}_t^{o_4}) = \text{MSDefAttn}(\mathcal{F}_t^{b_2}, \mathcal{F}_t^{b_3}, \mathcal{F}_t^{b_4}), \quad (1)$$

where MSDefAttn denotes multi-scale deformable attention, $\mathcal{F}_t^{o_i}$ denotes the i -th stage multi-scale features of t -th frame. Moreover, the FPN generates mask features \mathcal{F}_t^{mask} for each frame, providing reference features to the transformer decoder for mask generation.

Transformer Decoder. The main computational flow of transformer decoder begins by initializing a set of random queries $\mathcal{Q}_t^0 \in \mathbb{R}^{N \times d}$, where N denotes the number of queries. These queries are iteratively refined through M transformer decoder layers, producing instance queries $\mathcal{Q}_t^i \in \mathbb{R}^{N \times d}, i = 1, 2, 3, \dots, M$. The transformer decoder

layer refers to the masked attention transformer decoder from mask2former. These refined queries are then decoded into class logits $\mathcal{C}_t \in \mathbb{R}^{N \times C}$ for classification and mask logits $\mathcal{M}_t \in \{0, 1\}^{N \times H \times W}$ for segmentation. It is important to note that the queries at each layer are supervised, meaning the output of queries from any layer can effectively serve as the model’s final output.

Tracker. Trackers vary widely, and high-quality ones can greatly improve online VIS. However, these trackers often come with high computational costs. For example, CTVIS [48] employs mask non-maximum suppression (NMS) and a memory bank for instance tracking, but this introduces significant inference overhead. To mitigate this, we follow MinVIS and use the Hungarian algorithm [25] to match queries between adjacent frames, generating reordered video instance queries and masks. Note that, DVIS++ designs a referring tracker which is also based on Hungarian algorithm so that it is also efficient.

3.2. Temporal-aware Query Routing

To reduce redundancy in the transformer decoder, we propose TAR and an auxiliary Synchronizer module. During the t -th frame transformer decoder inference, after M^s lay-

ers, the initial random queries \mathcal{Q}_t^0 are refined into $\mathcal{Q}_t^{M^s}$, where M^s denotes the warm-up transformer decoder layers before synchronizer. $\mathcal{Q}_t^{M^s}$ are then fed into the Synchronizer to align with the final queries $\hat{\mathcal{Q}}_{t-1}^M$ from the previous frame. Before each transformer decoder layer, the synchronized queries $\hat{\mathcal{Q}}_t^{M^s}$ and $\hat{\mathcal{Q}}_{t-1}^M$ are fused in the TAR to determine whether to skip the current layer.

Synchronizer. To determine whether to skip layers by calculating the differences between queries across frames, we first need to align the current frame’s queries with those from previous frames to obtain meaningful differential information. However, since the initial queries are randomly initialized and lack semantic meaning, we allow them to pass through M^s layers before alignment. To facilitate easier optimization of the Synchronizer, we design the following approach:

$$\hat{\mathcal{Q}}_t^{M^s} = \text{Attention}(\hat{\mathcal{Q}}_{t-1}^M, \mathcal{Q}_t^{M^s}, \mathcal{Q}_t^{M^s}), \quad (2)$$

$$\hat{\mathcal{Q}}_t^{M^s} = \text{Norm}(\hat{\mathcal{Q}}_t^{M^s} + \hat{\mathcal{Q}}_{t-1}^M), \quad (3)$$

where Attention is the scaled-dot production attention [39], Norm denotes the LayerNorm [3], $\hat{\mathcal{Q}}_t^{M^s}$ is the synchronized queries of the current frame, $\hat{\mathcal{Q}}_{t-1}^M$ is the final queries of the previous frame. If $t = 0$, $\hat{\mathcal{Q}}_{-1}^M = \mathcal{Q}_0^M$.

Router. After acquiring the synchronized queries $\hat{\mathcal{Q}}_t^{M^s}$ of current frame, we concatenate them with the final queries $\hat{\mathcal{Q}}_{t-1}^M$ from the previous frame and their differential information. We then input the resulting combination into a multi-layer perceptron (MLP) for predicting a binary routing score:

$$\mathcal{S}_t^i = \text{MLP}(\text{concat}(\hat{\mathcal{Q}}_t^i, \hat{\mathcal{Q}}_{t-1}^M, \hat{\mathcal{Q}}_t^i - \hat{\mathcal{Q}}_{t-1}^M)), \quad (4)$$

$$s_t^i = \sum_{j=1}^N \mathcal{S}_t^{i,j}, \quad (5)$$

$$i \in \{M^s + 1, M^s + 2, \dots, M\}, \quad (6)$$

where concat operates along the channel dimension, $\text{MLP}(\cdot)$ is comprised of $\text{Linear}(\text{ReLU}(\text{Linear}(\cdot)))$, $\mathcal{S}_t^i \in \mathbb{R}^{N \times 2}$ denotes the fine-grained routing scores for the i -th transformer decoder layer of the t -th frame, and $s_t^i \in \mathbb{R}^2$ denotes the global routing scores for the i -th transformer decoder layer of the t -th frame. The binary score s_t^i directly determines whether to skip the i -th transformer decoder layer at the t -th frame:

$$\hat{\mathcal{Q}}_t^i = \begin{cases} \hat{\mathcal{Q}}_t^{i-1}, & \text{argmax}(s_t^i) = 0, \\ \text{TD}^i(\hat{\mathcal{Q}}_t^{i-1}), & \text{argmax}(s_t^i) = 1, \end{cases} \quad (7)$$

where argmax denotes selecting the index of the maximum value in the routing score vector s_t^i , TD^i denotes i -th transformer decoder layers. Furthermore, since argmax is non-differentiable during training, we use the straight-through

trick introduced in [38] as a replacement for argmax . The computation process is described below:

$$g_t^i = \text{softmax}(s_t^i), \quad (8)$$

$$\hat{g}_t^i = \text{argmax}(g_t^i) + g_t^i - \text{sg}(g_t^i), \quad (9)$$

where softmax denotes the softmax function that converts the routing scores into a probability distribution, sg denotes the stop-gradient operation that prevents gradients from flowing through its argument, $g_t^i \in \mathbb{R}^2$ denotes the softmax-normalized routing scores for the i -th layer of the t -th frame, $\hat{g}_t^i \in \{0, 1\}^2$ denotes the differentiable approximation of the one-hot encoded argmax output. In Equation 9, since g_t^i is differentiable and \hat{g}_t^i is constructed by adding two constants to g_t^i , \hat{g}_t^i is also differentiable. This property allows for a differentiable routing operation during training:

$$\hat{\mathcal{Q}}_t^i = \hat{\mathcal{Q}}_t^{i-1} \cdot \hat{g}_t^i[0] + \text{TD}^i(\hat{\mathcal{Q}}_t^{i-1}) \cdot \hat{g}_t^i[1]. \quad (10)$$

4. Experiments

4.1. Setup

Datasets. We evaluate our method on three challenging VIS datasets: YoutubeVIS 2019 [45], YoutubeVIS 2021 [46], and OVIS [34]. YoutubeVIS 2019 is the most widely used VIS dataset, consisting of 2238/302/343 videos with 40 object categories for the training/validation/testing sets, respectively. YoutubeVIS 2021 builds upon YoutubeVIS 2019, offering more videos and richer annotations, with 2985/421/453 videos for the training/validation/testing sets. OVIS is a highly complex dataset designed to address scenarios with heavy occlusion and a large number of objects. Although it has 607/140/154 videos for training/validation/testing with 25 semantic categories, with an average of 5.8 instances per video much higher than YoutubeVIS 2019. OVIS also features long video sequences, with some videos exceeding 500 frames, which is 13.5 times longer than those in YouTubeVIS 2019.

Metrics. To assess efficiency, we adopt metrics including Speed and FPS. For performance evaluation, we employ metrics including AP (Average Precision), AP₇₅ (Average Precision with an IoU threshold 0.75), and AR₁₀ (Average Recall with 10 detections per frame). AP₇₅ measures the precision of predictions with high localization accuracy, while AP and AR₁₀ evaluate overall detection accuracy.

Implementation Details. We primarily selected two baselines for experiments: MinVIS and DVIS++. For MinVIS, we loaded its pre-trained model, froze the image encoder, and inserted the Synchronizer and TAR. We trained the model on pairs of adjacent video frames using the AdamW optimizer with a learning rate of 1e-4 for 5,000 iterations,

Method	↓ Speed (ms)	↑ FPS	YoutubeVIS 2019			YoutubeVIS 2021		
			AP	AP ₇₅	AR ₁₀	AP	AP ₇₅	AR ₁₀
Small Backbone (Resnet50)								
MinVIS <small>[Neurips 2022]</small> [19]	40.5	24.7	47.4	52.1	55.7	44.2	48.1	51.7
DVIS <small>[ICCV 2023]</small> [51]	41.7	23.9	51.2	57.1	59.3	46.4	49.6	53.5
CTVIS <small>[ICCV 2023]</small> [48]	386.4	2.6	55.1	59.1	63.2	50.1	54.7	59.5
DVIS++ <small>[Arxiv 2312]</small> [52]	44.7	22.4	55.5	60.1	62.6	50.0	54.5	58.4
DVIS-DAQ <small>[ECCV 2024]</small> [53]	854.6	1.2	55.2	61.9	63.7	50.4	55.0	57.6
MinVIS <small>[Neurips 2022]</small> [19]	40.5	24.7	47.4	52.1	55.7	44.2	48.1	51.7
TAR+MinVIS	28.9 ↓ 11.6	34.6 ↑ 9.9	48.4 ↑ 1.0	53.1 ↑ 1.0	58.5 ↑ 2.8	44.3 ↑ 0.1	48.2 ↑ 0.1	53.0 ↑ 1.3
DVIS++ <small>[arxiv 2312]</small> [52]	44.7	22.4	55.5	60.1	62.6	50.0	54.5	58.4
TAR+DVIS++	30.5 ↓ 14.2	32.8 ↑ 10.4	55.7 ↑ 0.2	59.4 ↓ 0.7	63.7 ↑ 1.1	50.8 ↑ 0.8	53.7 ↓ 0.8	59.4 ↑ 1.0
Large Backbone (Swin-L / ViT-Adapter-L)								
MinVIS <small>[Neurips 2022]</small> [19]	63.2	15.8	61.6	68.6	66.6	55.3	62.0	60.8
DVIS <small>[ICCV 2023]</small> [51]	75.1	13.3	63.9	70.4	69.0	58.7	66.6	64.6
CTVIS <small>[ICCV 2023]</small> [48]	397.6	2.5	65.6	72.2	70.4	61.2	68.8	65.8
*DVIS++ <small>[Arxiv 2312]</small> [52]	135.9	7.4	67.7	75.3	73.7	62.3	70.2	68.0
DVIS-DAQ <small>[ECCV 2024]</small> [53]	>1000	<1	65.7	73.6	70.7	61.1	68.2	66.6
*DVIS-DAQ <small>[ECCV 2024]</small> [53]	>1000	<1	68.3	76.1	73.5	62.4	70.8	68.0
MinVIS <small>[Neurips 2022]</small> [19]	63.2	15.8	61.6	68.6	66.6	55.3	62.0	60.8
TAR+MinVIS	54.1 ↓ 9.1	18.5 ↑ 2.7	62.5 ↑ 0.9	68.7 ↑ 0.1	68.1 ↑ 1.5	57.2 ↑ 1.9	64.2 ↑ 2.2	63.4 ↑ 2.6
*DVIS++ <small>[Arxiv 2312]</small> [52]	135.9	7.4	67.7	75.3	73.7	62.3	70.2	68.0
*TAR+DVIS++	124.7 ↓ 11.2	8.0 ↑ 0.6	68.1 ↑ 0.4	75.0 ↓ 0.3	73.5 ↓ 0.2	62.7 ↑ 0.4	69.5 ↓ 0.7	68.3 ↑ 0.3

Table 1. **Main Results on YoutubeVIS 2019 [45] and YoutubeVIS 2021 [46] val split.** For small backbones, all models utilize ResNet50 [16]. For large backbones, most models employ Swin-Large [30], except for those marked with *, which indicates the use of DINOv2 [32]-based ViT-Adapter-L [9]. Inference speed and FPS are measured at a 480p video resolution on an NVIDIA 3090 GPU using frame-by-frame online inference. Except for MinVIS and TAR+MinVIS, all other models utilize additional data (i.e., COCO pseudo videos). Moreover, all of the evaluation results are measured at 480p, except for MinVIS and TAR+MinVIS, which are evaluated at 360p.

followed by 1,000 iterations at a reduced learning rate of $1e-5$. The batch size was set to 8, and the weight decay was $5e-2$. The training progress of MinVIS didn't use extra training data, e.g., COCO pseudo videos. For DVIS++, we initialized the segmenter with pre-trained weights from CTVIS. During the online training phase, we froze the segmenter and integrated the Synchronizer and router into it. The remaining training settings were consistent with DVIS++, including the frame sampling strategy (5 consecutive video frames), optimizer configuration (AdamW with a weight decay of $5e-2$), and training schedule (20,000 iterations for OVIS and 40,000 iterations for YouTubeVIS), learning rate schedule (The learning rate was initially set to $1e-4$ and decayed to $1e-5$ at 14,000 iterations for OVIS and 28,000 iterations for YouTubeVIS). The training progress of DVIS++ used COCO extra data during YoutubeVIS training, but not used COCO extra data during OVIS training.

4.2. Main Results

YoutubeVIS 2019/2021. Based on the Table 1, we can conclude that **Our TAR can enhance the efficiency while maintaining comparable performance.** For instance, with a small backbone configuration, TAR combined with the baseline MinVIS (i.e., TAR+MinVIS) reduces the inference time from 40.5 ms to 28.9 ms, increases the FPS from 24.7 to 34.6, and improved the AP from 47.4/44.2 to 48.4/44.3 on YoutubeVIS 2019 and YoutubeVIS 2021. Additionally, TAR with the baseline DVIS++ (i.e., TAR+DVIS++) also significantly enhances efficiency while maintaining comparable performance. For large backbone, our TAR achieves similar results. For example, when combined with the baseline MinVIS (i.e., TAR + MinVIS), it reduces the inference time from 63.2 ms to 54.1 ms and increases the FPS from 15.8 to 28.5, resulting in performance improve-

Method	AP	AP ₇₅	AR ₁₀
Small Backbone (Resnet50)			
MinVIS <small>[Neurips 2022]</small> [19]	25.0	24.0	29.7
DVIS <small>[ICCV 2023]</small> [51]	30.2	30.5	37.3
CTVIS <small>[ICCV 2023]</small> [48]	35.5	34.9	41.9
DVIS++ <small>[Arxiv2312]</small> [52]	37.2	37.3	42.9
DVIS-DAQ <small>[ECCV 2024]</small> [53]	38.7	37.6	45.2

MinVIS <small>[Neurips 2022]</small> [19]	25.0	24.0	29.7
TAR+MinVIS	27.4 ↑ 2.4	27.1 ↑ 3.1	33.2 ↑ 3.5

DVIS++ <small>[Arxiv2312]</small> [52]	37.2	37.3	42.9
TAR+DVIS++	38.1 ↑ 0.9	36.2 ↓ 0.9	43.9 ↑ 1.0

Large Backbone (Swin-L / ViT-Adapter-L)			
MinVIS <small>[Neurips2022]</small> [19]	39.4	41.3	43.3
DVIS <small>[ICCV 2023]</small> [51]	45.9	48.3	51.5
CTVIS <small>[ICCV 2023]</small> [48]	46.9	47.5	52.1
*DVIS++ <small>[Arxiv2312]</small> [52]	49.6	55.0	54.6
DVIS-DAQ <small>[ECCV 2024]</small> [53]	49.5	51.7	54.9

MinVIS <small>[Neurips 2022]</small> [19]	39.4	41.3	43.3
TAR+MinVIS	40.6 ↑ 1.2	42.9 ↑ 1.6	45.3 ↑ 2.0

*DVIS++ <small>[Arxiv2312]</small> [52]	49.6	55.0	54.6
*TAR+DVIS++	50.2 ↑ 0.6	54.2 ↓ 0.8	55.4 ↑ 0.8

Table 2. **Main Results on OVIS** [34] *val* split. This set of experiments is aligned with the setup in Table 1.

ments on both the YouTubeVIS 2019 and YouTubeVIS 2021 datasets. Similarly, TAR+DVIS++ also enhances the efficiency of DVIS++ (reducing from 135.9 ms to 124.7 ms) while maintaining performance consistency.

OVIS. From the results in Table 2, we observe that our method remains effective under long video evaluation settings. With both small and large backbones, our approach achieves comparable or slightly better performance compared to the two baselines (i.e., MinVIS and DVIS++). Additionally, it is worth noting that the efficiency improvements of our method on OVIS are consistent with those on YoutubeVIS (i.e., Table 1).

4.3. Ablation Study

Design Choices Ablation. Based on the results in Table 3, we draw two main conclusions: (1) **The Necessity of the Synchronizer for the Router.** Without the Synchronizer aligning queries between the current and previous frames, the Router loses its intended functionality. It not only fails to deliver significant efficiency gains (only a 2.8 FPS improvement) but also causes a substantial drop in the original model’s performance (47.4 → 45.1 mAP). (2) **Cross Atten-**

Synchronizer	Router	FPS	mAP	AR ₁₀
MinVIS		24.7	47.4	55.7
	✓	27.5 ↑ 2.8	45.1 ↓ 2.3	52.3 ↓ 3.4
Bipartite		23.3 ↓ 1.4	45.8 ↓ 1.6	54.5 ↓ 1.2
	✓	31.3 ↑ 6.6	46.4 ↓ 1.0	55.5 ↓ 0.2
Cross Attn.		24.2 ↓ 0.5	47.9 ↑ 0.5	57.2 ↑ 1.5
	✓	34.6 ↑ 9.9	48.4 ↑ 1.0	58.5 ↑ 2.8

Table 3. **Ablation Study** of our designs (Synchronizer and Router) for TAR. The results is evaluated on YoutubeVIS 2019 *val* split. “Bipartite” refers to the use of bipartite graph matching to synchronize queries across adjacent frames.

Model	FPS	mAP	AR ₁₀
MinVIS	24.7	47.4	55.7
+ flash attention 2 [13]	27.8 ↑ 3.1	47.9 ↑ 0.5	55.3 ↓ 0.4
+ operation fusion	27.2 ↑ 2.5	47.4	55.6 ↓ 0.1
+ TAR	31.7 ↑ 7.0	48.5 ↑ 1.1	58.9 ↑ 3.2
+ all 3 operations	34.6 ↑ 9.9	48.4 ↑ 1.0	58.5 ↑ 2.8

Table 4. **Ablation Study** of TAR and other auxiliary operations on speed and performance.

tion is a more effective Synchronizer for the Router. For instance, when bipartite graph matching is used as the Synchronizer, efficiency improves (24.7 → 31.3 FPS), but performance declines (47.4 → 46.4 mAP). In contrast, Cross Attention, a learnable Synchronizer, inherently boosts performance and, when combined with the Router, further enhances both performance and efficiency.

Implementation Ablation. Building on the standalone introduction of TAR, we identify further optimization opportunities in existing online VIS implementations. As shown in Table 4, integrating Flash Attention 2 [13] and optimizing PyTorch operations also improve inference speed (e.g., 24.7 → 27.8 and 24.7 → 27.2, respectively) with minimal performance impact. When combined with TAR, these optimizations further enhance efficiency (31.7 FPS → 34.6 FPS).

Hyperparameters Ablation. We ablate M^s (i.e., the number of warm-up transformer decoder layers) in Figure 4. The parameter M^s also determines the routing search space between skipping and non-skipping layers, which amounts to 2^{M-M^s} . From the results in the figure, we observe a trend: as M^s increases, the search space 2^{M-M^s} decreases, leading to a decline in both performance and efficiency. Setting $M^s = 1$ maximizes the search space, achieving the best balance of performance and efficiency.

4.4. Discussion

Does the transformer decoder really have redundant layers? In Table 5, we provide a detailed analysis of the

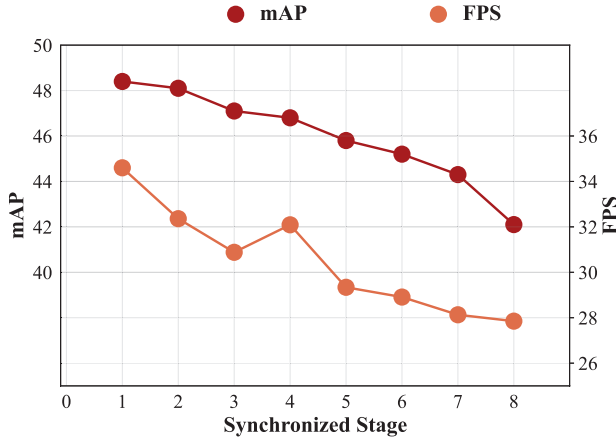


Figure 4. Effect of M^s (i.e., the number of warm-up transformer decoder layers before Synchronizer) on FPS and mAP.

Model	Layers	FPS	mAP	AR ₁₀
MinVIS	9	24.7	47.4	55.7
MinVIS	1	41.4 $\uparrow 16.7$	37.9 $\downarrow 9.5$	48.2 $\downarrow 7.5$
MinVIS	2	38.5 $\uparrow 13.8$	43.2 $\downarrow 4.2$	53.6 $\downarrow 2.1$
MinVIS	3	36.5 $\uparrow 11.8$	44.9 $\downarrow 2.5$	53.7 $\downarrow 2.0$
MinVIS	6	29.7 $\uparrow 5.0$	47.6 $\uparrow 0.2$	55.2 $\downarrow 0.5$
TAR+MinVIS	adaptive	34.6 $\uparrow 9.8$	48.4 $\uparrow 1.0$	58.5 $\uparrow 2.8$

Table 5. Inference speed and performance of MinVIS with varying numbers of layers, and MinVIS after integrating TAR.

speed and performance of the trained MinVIS model with varying numbers of layers. A key insight emerges: **selecting an proper number of inference layers can achieve higher efficiency and performance compared to using all layers**. For instance, using 6 layers improves both inference efficiency (24.7 \rightarrow 29.7 FPS) and overall performance (47.4 \rightarrow 47.6 mAP). Furthermore, TAR enables searching across the 2^{M-M^s} possible layer combinations, learning the optimal number of layers for different samples during training to maximize redundancy reduction. By combining insights from Figure 2 and Table 5, we conclude that redundancy exists within the transformer decoder layers.

What determines the number of layers skipped by the transformer decoder? In Figure 5, we train the model using hqytvis [23] and analyze the correlation distribution of adjacent frame RGB MSE along with the number of skipped transformer decoder layers on the *val* split of hqytvis. This reveals a key insight: **the smaller the inter-frame RGB MSE, the lower the redundancy in the transformer decoder, and the fewer layers need to be skipped**. This aligns with our discussion in the penultimate paragraph of Section 1. Figure 5 demonstrates that TAR can determine

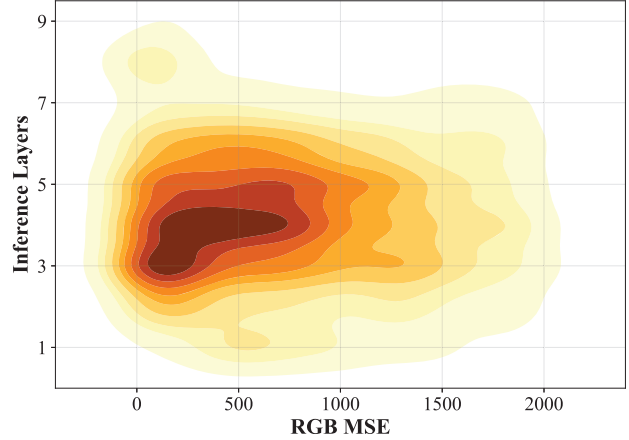


Figure 5. Correlation between inter-frame RGB MSE and the number of non-skipped transformer decoder layers during inference, measured on hqytvis *val*.

whether to skip transformer decoder layers based on inter-frame differences. Smaller inter-frame differences indicate that TAR considers fewer layers necessary to achieve results comparable to the previous frame.

5. Conclusion

In this work, we identify redundant computations in transformer decoders for online VIS, which hinder inference efficiency. Analyzing output similarities between adjacent frames across decoder layers reveals substantial redundancy, leading us to propose TAR that dynamically skips redundant layers using inter-frame information. Experimental results show integrating TAR into online VIS baselines (MinVIS, DVIS++) boosts inference speed (24.7 \rightarrow 34.6 FPS, 22.4 \rightarrow 34.8 FPS) while maintaining or enhancing performance (e.g., 1.0/0.2 mAP on YouTubeVIS 2019, 0.1/0.8 mAP on YoutubeVIS 2021, 2.4/0.9 mAP on OVIS). Furthermore, our analysis reveals that the number of skipped layers increases as the differences between adjacent frames decrease, validating that TAR effectively utilizes inter-frame differences to reduce redundant computations. In summary, TAR provides a practical and efficient solution for online VIS, enabling faster inference without compromising accuracy. This work introduces a new technical route, i.e., using conditional computation for optimizing the efficiency of transformer-based online VIS methods.

Acknowledgements. This work was supported in part by the Shenzhen Medical Research Funds in China (No. B2302037), National Natural Science Foundation of China (NSFC) under Grant No. 61972217, 32071459, 62176249, 62006133, 62271465, 62406167, U24B6012 and AI for Science (AI4S)-Preferred Program, Peking University Shenzhen Graduate School, China.

References

- [1] Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, et al. Colt5: Faster long-range transformers with conditional computation. *arXiv preprint arXiv:2303.09752*, 2023. 3
- [2] Ali Athar, Sabarinath Mahadevan, Aljosa Osep, Laura Leal-Taixé, and Bastian Leibe. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *Proceedings of the European Conference on Computer Vision*, pages 158–177. Springer, 2020. 3
- [3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5
- [4] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015. 3
- [5] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3
- [6] Gedas Bertasius and Lorenzo Torresani. Classifying, segmenting, and tracking object instances in video with mask propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9739–9748, 2020. 3
- [7] Jiale Cao, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. Sipmask: Spatial information preservation for fast image and video instance segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 1–18. Springer, 2020. 3
- [8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergej Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision*, pages 213–229. Springer, 2020. 3
- [9] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022. 6
- [10] Bowen Cheng, Anwesa Choudhuri, Ishan Misra, Alexander Kirillov, Rohit Girdhar, and Alexander G Schwing. Mask2former for video instance segmentation. *arXiv preprint arXiv:2112.10764*, 2021. 3
- [11] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022. 1, 4
- [12] Raj Dabre, Raphael Rubino, and Atsushi Fujita. Balancing cost and benefit with tied-multi transformers. *arXiv preprint arXiv:2002.08614*, 2020. 3
- [13] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. 7
- [14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3
- [15] Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*, 2019. 3
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6
- [17] Miran Heo, Sukjun Hwang, Seoung Wug Oh, Joon-Young Lee, and Seon Joo Kim. Vita: Video instance segmentation via object token association. 2022. 1
- [18] Miran Heo, Sukjun Hwang, Jeongseok Hyun, Hanjung Kim, Seoung Wug Oh, Joon-Young Lee, and Seon Joo Kim. A generalized framework for video instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14623–14632, 2023. 1, 3
- [19] De-An Huang, Zhiding Yu, and Anima Anandkumar. Minvis: A minimal video instance segmentation framework without video-based training. *arXiv preprint arXiv:2208.02245*, 2022. 3, 6, 7
- [20] Shihua Huang, Zhichao Lu, Ran Cheng, and C FaPN He. Feature-aligned pyramid network for dense image prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 864–873, 2021. 4
- [21] Sukjun Hwang, Miran Heo, Seoung Wug Oh, and Seon Joo Kim. Video instance segmentation using inter-frame communication transformers. *Advances in Neural Information Processing Systems*, 34:13352–13363, 2021. 3
- [22] Lei Ke, Henghui Ding, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Video mask transfiner for high-quality video instance segmentation. In *European Conference on Computer Vision*, pages 731–747. Springer, 2022. 3
- [23] Lei Ke, Henghui Ding, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Video mask transfiner for high-quality video instance segmentation. In *European Conference on Computer Vision (ECCV)*, 2022. 8
- [24] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6399–6408, 2019. 4
- [25] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 3, 4
- [26] Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Zhao, Yuexin Wu, Bo Li, et al. Conditional adapters: Parameter-efficient transfer learning with fast inference. *Advances in Neural Information Processing Systems*, 36:8152–8172, 2023. 3
- [27] Minghan Li, Shuai Li, Lida Li, and Lei Zhang. Spatial feature calibration and temporal fusion for effective one-stage video instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11215–11224, 2021. 3

- [28] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 4
- [29] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178*, 2020. 3
- [30] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 6
- [31] Tim Meinhardt, Matt Feiszli, Yuchen Fan, Laura Leal-Taixe, and Rakesh Ranjan. Novis: A case for end-to-end near-online video instance segmentation. *arXiv preprint arXiv:2308.15266*, 2023. 1, 3
- [32] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 6
- [33] Kwanyong Park, Sanghyun Woo, Seoung Wug Oh, In So Kweon, and Joon-Young Lee. Mask-guided matting in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1992–2001, 2023. 3
- [34] Jiyang Qi, Yan Gao, Yao Hu, Xinggang Wang, Xiaoyu Liu, Xiang Bai, Serge Belongie, Alan Yuille, Philip Torr, and Song Bai. Occluded video instance segmentation: A benchmark. *International Journal of Computer Vision*, 2022. 5, 7
- [35] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022. 3
- [36] Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A Smith. The right tool for the job: Matching model and instance complexities. *arXiv preprint arXiv:2004.07453*, 2020. 3
- [37] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 4
- [38] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 5
- [39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 5
- [40] Xin Wang, Yujia Luo, Daniel Crankshaw, Alexey Tumanov, Fisher Yu, and Joseph E Gonzalez. Idk cascades: Fast deep learning by learning not to overthink. *arXiv preprint arXiv:1706.00885*, 2017. 3
- [41] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8741–8750, 2021. 3
- [42] Junfeng Wu, Yi Jiang, Song Bai, Wenqing Zhang, and Xiang Bai. Seqformer: Sequential transformer for video instance segmentation. In *European Conference on Computer Vision*, pages 553–569. Springer, 2022. 3
- [43] Junfeng Wu, Qihao Liu, Yi Jiang, Song Bai, Alan Yuille, and Xiang Bai. In defense of online models for video instance segmentation. In *European Conference on Computer Vision*, pages 588–605. Springer, 2022. 3
- [44] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020. 3
- [45] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5188–5197, 2019. 1, 3, 5, 6
- [46] Linjie Yang, Yuchen Fan, Yang Fu, and Ning Xu. The 3rd large-scale video object segmentation challenge - video instance segmentation track, 2021. 5, 6
- [47] Shusheng Yang, Xinggang Wang, Yu Li, Yuxin Fang, Jiemin Fang, Wenyu Liu, Xun Zhao, and Ying Shan. Temporally efficient vision transformer for video instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2885–2895, 2022. 3
- [48] Kaining Ying, Qing Zhong, Weian Mao, Zhenhua Wang, Hao Chen, Lin Yuanbo Wu, Yifan Liu, Chengxiang Fan, Yunzhi Zhuge, and Chunhua Shen. Ctvts: Consistent training for online video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 899–908, 2023. 4, 6, 7
- [49] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2020. 3
- [50] Dewen Zeng, Nan Du, Tao Wang, Yuanzhong Xu, Tao Lei, Zhifeng Chen, and Claire Cui. Learning to skip for language modeling. *arXiv preprint arXiv:2311.15436*, 2023. 3
- [51] Tao Zhang, Xingye Tian, Yu Wu, Shunping Ji, Xuebo Wang, Yuan Zhang, and Pengfei Wan. Dvis: Decoupled video instance segmentation framework. *arXiv preprint arXiv:2306.03413*, 2023. 3, 6, 7
- [52] Tao Zhang, Xingye Tian, Yikang Zhou, Shunping Ji, Xuebo Wang, Xin Tao, Yuan Zhang, Pengfei Wan, Zhongyuan Wang, and Yu Wu. Dvis++: Improved decoupled framework for universal video segmentation. *arXiv preprint arXiv:2312.13305*, 2023. 3, 6, 7
- [53] Yikang Zhou, Tao Zhang, Shunping Ji, Shuicheng Yan, and Xiangtai Li. Improving video segmentation via dynamic anchor queries. In *European Conference on Computer Vision*, pages 446–463. Springer, 2024. 6, 7
- [54] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *Proceedings of the International Conference on Learning Representations*, 2020. 4