

MobileInst: Video Instance Segmentation on the Mobile

Renhong Zhang^{1*}, Tianheng Cheng^{1*}, Shusheng Yang¹, Haoyi Jiang¹, Shuai Zhang², Jiancheng Lyu², Xin Li², Xiaowen Ying², Dashan Gao², Wenyu Liu¹, Xinggong Wang^{1†}

¹ School of EIC, Huazhong University of Science & Technology

² Qualcomm AI Research, Qualcomm Technologies, Inc

Abstract

Video instance segmentation on mobile devices is an important yet very challenging edge AI problem. It mainly suffers from (1) heavy computation and memory costs for frame-by-frame pixel-level instance perception and (2) complicated heuristics for tracking objects. To address these issues, we present MobileInst, a lightweight and mobile-friendly framework for video instance segmentation on mobile devices. Firstly, MobileInst adopts a mobile vision transformer to extract multi-level semantic features and presents an efficient query-based dual-transformer instance decoder for mask kernels and a semantic-enhanced mask decoder to generate instance segmentation per frame. Secondly, MobileInst exploits simple yet effective kernel reuse and kernel association to track objects for video instance segmentation. Further, we propose temporal query passing to enhance the tracking ability for kernels. We conduct experiments on COCO and YouTube-VIS datasets to demonstrate the superiority of MobileInst and evaluate the inference latency on one single CPU core of the Snapdragon 778G Mobile Platform, without other methods of acceleration. On the COCO dataset, MobileInst achieves 31.2 mask AP and 433 ms on the mobile CPU, which reduces the latency by 50% compared to the previous SOTA. For video instance segmentation, MobileInst achieves 35.0 AP and 30.1 AP on YouTube-VIS 2019 & 2021.

Introduction

Deep visual understanding algorithms with powerful GPUs have achieved great success, but their performance is reaching a plateau. Edge AI, which enables massive low-resource computing devices, is becoming increasingly popular. In this paper, we study a very challenging edge AI task, namely video instance segmentation (VIS) on mobile devices. The goal of VIS (Yang, Fan, and Xu 2019) is to simultaneously identify, segment, and track objects in the video sequence and it attracts a wide range of applications, *e.g.*, robotics, autonomous vehicles, video editing, and augmented reality. The advances in deep convolutional neural networks and vision transformers have made great progress in video instance

*These authors contributed equally.

†Xinggong Wang (xgwang@hust.edu.cn) is the corresponding author.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

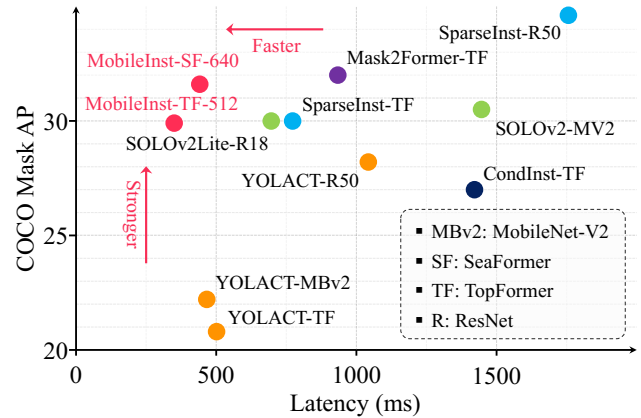


Figure 1: Speed-and-Accuracy Trade-off. We evaluate all models on COCO *test-dev* and inference speeds are measured on one mobile CPU, *i.e.*, Snapdragon 778G. The proposed MobileInst outperforms other methods in both speed and accuracy on mobile devices.

segmentation and achieved tremendous performance (Bertasi and Torresani 2020; Athar et al. 2020; Lin et al. 2021) on GPUs. Nevertheless, many real-world applications tend to require those VIS methods to run on resource-constrained devices, *e.g.*, mobile phones, and inference with low latency. It’s challenging but urgent to develop and deploy efficient approaches for VIS on mobile or embedded devices.

Albeit great progress has been witnessed in the VIS field, there are several obstacles that prevent modern VIS frameworks from being deployed on edge devices with limited resources, such as mobile chipsets. Prevalent methods for video instance segmentation can be categorized into two groups: offline methods (clip-level) and online methods (frame-level). Offline methods (Wang et al. 2021b; Hwang et al. 2021; Yang et al. 2022; Wu et al. 2022a; Heo et al. 2022; Lin et al. 2021) divide the video into clips, generate the instance predictions for each clip, and then associate the instances by instance matching across clips. However, inference with clips (multiple frames) is infeasible in mobile devices in terms of computation and memory cost. Whereas, online methods (Yang, Fan, and Xu 2019; Yang et al. 2021; Cao et al. 2020; Fu et al. 2020; Wu et al. 2022b)

forward and predict with frame-level input but require complicated heuristic procedures to associate instances across frames, *e.g.*, NMS, which are inefficient in mobile devices. In addition, recent methods for video instance segmentation tend to employ heavy architectures, especially for the methods based on transformers, which incur a large computation burden and memory costs. Directly scaling down the model size for lower inference latency will inevitably cause severe performance degradation, which limits the practical application of recent methods. Designing and deploying video instance segmentation techniques for resource-constrained devices have not been well explored yet, which are not trivial but crucial for real-world applications.

In this paper, we introduce MobileInst to achieve performant video instance segmentation on mobile devices for the first time. MobileInst is efficient and mobile-friendly from two key aspects (1) lightweight architectures for segmenting objects per frame and (2) simple yet effective temporal modeling for tracking instances across frames. Specifically, MobileInst consists of a query-based *dual transformer instance decoder*, which exploits object queries to segment objects, updates object queries through global contexts and local details, and then generates the mask kernels and classification scores. To efficiently aggregate multi-scale features and global contexts for mask features, MobileInst employs a *semantic-enhanced mask decoder*. The object queries are forced to represent objects in a one-to-one manner and we discover that mask kernels (generated by object queries) tend to be *temporally consistent* in consecutive frames, *i.e.*, the same kernel (query) corresponds to the same objects in nearby frames, as shown in Fig. 2. Therefore, we exploit simple yet effective *kernel reuse* and *kernel association* to track objects by reusing kernels in a T -frame clips and associate objects across clips by kernel cosine similarity. Further, we present temporal query passing to enhance the tracking ability for object queries during training with video sequences. MobileInst can one-the-fly segment and track objects in videos on mobile devices.

The main contributions can be summarized as follows:

- We present a cutting-edge and mobile-friendly framework named MobileInst for video instance segmentation on mobile devices, which is the first work targeting VIS on mobile devices to the best of our knowledge.
- We propose a dual transformer instance decoder and semantic-enhanced mask decoder in MobileInst for efficiently segmenting objects in frames.
- We present kernel reuse and kernel association for tracking objects across frames which are simple and efficient along with the temporal training strategy.
- We benchmark the mobile VIS problem by implementing a wide range of lightweight VIS methods for comparisons. The proposed MobileInst can achieve state-of-the-art mobile VIS performance, *i.e.*, 35.0 AP with 188 ms on YouTube-VIS-2019 (Yang, Fan, and Xu 2019) and 31.2 AP with 433 ms on COCO (Lin et al. 2014) *test-dev*, when deployed on the CPU of Snapdragon 778G, without using mixed precision, low-bit quantization, or the inside hardware accelerator for neural network inference.

Related Work

Instance Segmentation

Most methods address instance segmentation by extending object detectors with mask branches, *e.g.*, Mask R-CNN (He et al. 2017) adds an RoI-based fully convolutional network upon Faster R-CNN (Ren et al. 2017) to predict object masks. (Tian, Shen, and Chen 2020; Bolya et al. 2019; Xie et al. 2020; Zhang et al. 2020) present single-stage methods for instance segmentation. Several methods (Wang et al. 2020a,b; Cheng et al. 2022b) present detector-free instance segmentation for simplicity and efficiency. Recently, query-based detectors (Carion et al. 2020; Zhu et al. 2021; Fang et al. 2021b; Cheng, Schwing, and Kirillov 2021; Fang et al. 2021a) reformulate object detection with set prediction and show promising results on instance segmentation. Considering the inference speed, YOLACT (Bolya et al. 2019) and SparseInst (Cheng et al. 2022b) propose real-time methods and achieve a good trade-off between speed and accuracy. However, existing methods are still hard to deploy to mobile devices for practical applications due to the large computation burden and complex post-processing procedures.

Video Instance Segmentation

Offline Methods. Several methods (Wang et al. 2021b; Hwang et al. 2021; Yang et al. 2022; Wu et al. 2022a; Heo et al. 2022) take a video clip as the input once, achieving good performance due to the rich temporal information. VisTR (Wang et al. 2021b) proposes the first transformer-based offline VIS framework. Several works effectively alleviate the computation burden brought by self-attention by building Inter-frame Communication Transformers (Hwang et al. 2021), using messengers to exchange temporal information in the backbone (Yang et al. 2022), and focusing on temporal interaction of instance between frames (Wu et al. 2022a; Heo et al. 2022). However, clip-level input is difficult to apply to resource-constrained mobile devices.

Online Methods. Previous methods (Yang, Fan, and Xu 2019; Yang et al. 2021; Han et al. 2022) address online VIS by extending CNN-based image segmentation models to handle temporal coherence with extra embeddings to identify instances and associate instances with heuristic algorithms. However, those methods require extra complex post-processing steps, *e.g.*, NMS, which hinders end-to-end inference on mobile devices. Recently, transformer-based models address VIS by using simple tracking heuristics with object queries which have capabilities of distinguishing instances (Huang, Yu, and Anandkumar 2022). IDOL (Wu et al. 2022b) obtains performance comparable to offline VIS by contrastive learning of the instance embedding across frames. InsPro (He et al. 2023a) and InstanceFormer (Koner et al. 2023) respectively use proposals and reference points to establish correspondences between instances for online temporal propagation. Unfortunately, existing works rely on large-scale models like Mask2Former (Cheng et al. 2022a) and Deformable DETR (Zhu et al. 2021) beyond the capabilities of many mobile devices.



Figure 2: Reusing kernels for tracking. We train MobileInst for single-frame instance segmentation on YouTube-VIS 2019, and then apply MobileInst to infer the per-frame segmentation and track objects via reusing mask kernels. The upper row: We adopt the predicted mask kernel in $T = 1$ frame to obtain the segmentation results in the video sequence. In a short time, the reused mask kernels provide accurate segmentation and tracking results. The bottom row: We divide the videos into K -frame clips and reuse the mask kernels of every first frame. In addition, we adopt simple yet effective cosine similarity to associate the kernels in consecutive clips (K is set to 3). Reusing kernels with association performs well and is efficient.

Mobile Vision Transformers

Vision transformers (ViT) (Dosovitskiy et al. 2021) have demonstrated immense power in various vision tasks. Subsequent works (Liu et al. 2021b; Wang et al. 2021a; Fang et al. 2022) adopt hierarchical architectures and incorporate spatial inductive biases or locality into vision transformers for better feature representation. Vision transformers tend to be resource-consuming compared to convolutional networks due to the multi-head attention (Vaswani et al. 2017). To facilitate the mobile applications, MobileViT (Mehta and Rastegari 2022), MobileFormer (Chen et al. 2021) and TopFormer (Zhang et al. 2022) design mobile-friendly transformers by incorporating efficient transformer blocks into MobileNetV2 (Sandler et al. 2018). Recently, Wan et al. propose SeaFormer (Wan et al. 2023) with efficient axial attention. In this paper, MobileInst aims for video instance segmentation on mobile devices, which is more challenging than designing mobile backbones.

MobileInst

Overall Architecture

We present MobileInst, a video instance segmentation framework tailor-made for mobile devices. Fig. 3 gives an illustration of our framework. Given input images, MobileInst firstly utilizes a mobile transformer backbone to extract multi-level pyramid features. Following (Zhang et al. 2022; Wan et al. 2023), our backbone network consists of a series of convolutional blocks and transformer blocks. It takes images as inputs and generates both local features (*i.e.*, X_3 , X_4 , and X_5 in Fig. 3) and global features (*i.e.*, X_6).

Considering the global features X_6 contain abundant high-level semantic information, we present (1) *dual transformer instance decoder* which adopts a query-based transformer decoder based on the global image features and local image features and generates the instance predictions, *i.e.*, instance kernels and classification scores; (2) *semantic-enhanced mask decoder* which employs the multi-scale features from the backbone and a *semantic enhancer* to enrich the multi-scale features with semantic information.

Dual Transformer Instance Decoder

Queries are good trackers. Detection transformers with a sparse set of object queries (Carion et al. 2020) can get rid of heuristic post-processing for duplicate removal. Previous methods (Yang, Fan, and Xu 2019; Yang et al. 2021) extend dense detectors (Ren et al. 2017; Lin et al. 2017b; Tian et al. 2022) for VIS by designing heuristic matching to associate instances across frames, which is inefficient and hard to optimize in mobile devices. Whereas, as shown in Fig. 2, *object queries are good trackers* and can be used to associate objects in videos based on three reasons: (1) object queries are trained to segment the foreground of corresponding visual instance, thus naturally comprising contextualized instance features; (2) object queries are forced to match objects in a one-to-one manner and duplicate queries are suppressed; (3) the object query tends to be temporally consistent and represents the same instance in consecutive frames, which can be attributed to the temporal smoothness in adjacent frames. Therefore, using object queries as trackers can omit complex heuristics post-process for associating objects and is more efficient on mobile devices.

However, directly attaching transformer decoders like (Carion et al. 2020) on the mobile backbone leads to unaffordable computation budgets for mobile devices, and simply reducing decoder layers or parameters leads to unsatisfactory performance. Striking the balance and designing mobile-friendly architectures is non-trivial and critical for real-world applications. For efficiency, we present *dual transformer instance encoder*, which simplifies the prevalent 6-stage decoders in (Carion et al. 2020; Zhu et al. 2021) into 2-stage dual decoders, *i.e.*, the global instance decoder and the local instance decoder, which takes the global features X_G and local features X_L as key and value for updating object queries. We follow (Cheng et al. 2022a) and adopt the sine position embedding for both global and local features. The object queries Q are learnable and random initialized.

Global and Local Instance Decoder. Adding transformer encoders (Carion et al. 2020; Zhu et al. 2021) for the global contexts will incur a significant computation burden. In-

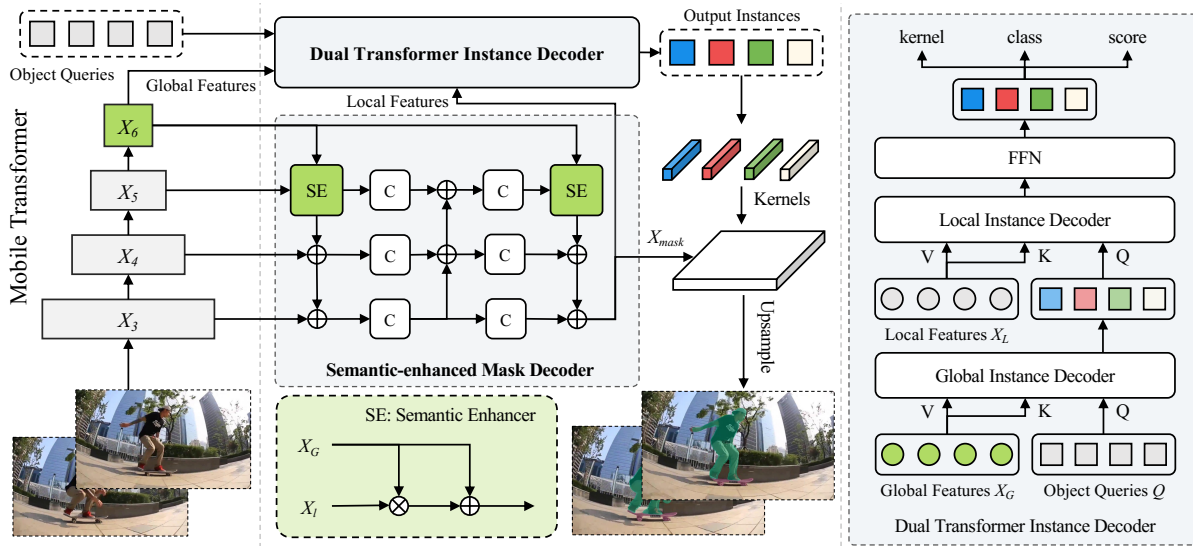


Figure 3: Overall architecture of MobileInst. MobileInst contains a mobile transformer as the backbone, a dual transformer instance decoder with learnable object queries to obtain object classes and kernels (Sec.), and a semantic-enhanced mask decoder to obtain single-level features of high-semantics (Sec.) by the semantic enhancers with global features X_G (X_6 from the mobile transformer). The generated kernels from instance queries and mask features X_{mask} can directly output the instance masks through the dot product. ‘C’ in the square denotes 3×3 convolution.

stead, we adopt high-level features (X_6 in Fig. 3) X_6 as global features X_G for query update, which contains high-level semantics and coarse localization. Inspired by recent works (Cheng, Schwing, and Kirillov 2021), we adopt the fine-grained local features, *i.e.*, the mask features X_{mask} , to compensate for spatial details for generating mask kernels. For efficiency, we downsample the mask features to $\frac{1}{64} \times$ through *max pooling*, *i.e.*, $X_L = f_{pool}(X_{mask})$, which can preserve more details. The *dual transformer instance decoder* acquires contextual features from the global features X_G and refines queries with fine-grained local features X_L .

Semantic-enhanced Mask Decoder

Multi-scale features are important for instance segmentation due to the severe scale variation in natural scenes. In addition, generating masks requires high-resolution features for accurate localization and segmentation quality. To this end, prevalent methods (Cheng, Schwing, and Kirillov 2021; Cheng et al. 2022a) stack multi-scale transformers (Cheng et al. 2022a) as pixel decoders to enhance the multi-scale representation and generate high-resolution mask features. Stacking transformers for high-resolution features leads to large computation and memory costs. Instead of using transformers, (Cheng et al. 2022b) presents a FPN-PPM encoder with 4 consecutive 3×3 convolutions as mask decoder, which also leads to a huge burden, *i.e.*, 7.6 GFLOPs. For mobile devices, we thus present an efficient *semantic-enhanced mask decoder*, as shown in Fig. 3. The mask decoder adopts the multi-scale features $\{X_3, X_4, X_5\}$ and outputs single-level high-resolution mask features ($\frac{1}{8} \times$). Motivated by FPN (Lin et al. 2017a), we use iterative *top-down* and *bottom-up* multi-scale fusion. Furthermore, we present

the *semantic enhancers* to strengthen the contextual information for the mask features with the global features X_6 , as shown in the green blocks of Fig. 3. Then the mask features X_{mask} and the generated kernels K are fused by $M = K \cdot X_{mask}$ to obtain the output segmentation masks.

Tracking with Kernel Reuse and Association

As discussed in Sec. , mask kernels (generated by object queries) are temporally consistent due to the temporal smoothness in adjacent frames. Hence, mask kernels can be directly adopted to segment and track the same instance in the nearby frames, *e.g.*, 11 frames as shown in Fig. 2. We thus present the efficient *kernel reuse* to adopt the mask kernels from the keyframe to generate the segmentation masks for the consecutive $T - 1$ frames as follows:

$$\begin{aligned} M^t &= K^t \cdot X_{mask}^t, \\ M^{t+i} &= K^t \cdot X_{mask}^{t+i}, i \in (0, \dots, T - 1), \end{aligned} \quad (1)$$

where $\{M^i\}_{i=t}^{T-1+t}$ are the segmentation masks for the same instance in the T -frame clip, and K^t is the reused mask kernel. Compared to clip-based methods, kernel reuse performs on-the-fly segmentation and tracking given per-frame input.

However, kernel reuse tends to fail in long-time sequences or frames with drastic changes. To remedy these issues, we follow (Huang, Yu, and Anandkumar 2022) and present a simple yet effective *kernel association*, which uses cosine similarity between the consecutive keyframes. Under one-to-one correspondence, duplicate queries (kernels) tend to be suppressed, which enables simple similarity metrics to associate kernels of consecutive keyframes. Compared to previous methods (Yang, Fan, and Xu 2019; Yang et al. 2021)

based on sophisticated metrics and post-processing methods, the proposed kernel association is much simple and easy to deploy on mobile devices.

MobileInst can be straightforwardly extended to video instance segmentation by incorporating the presented kernel reuse and association. And experimental results indicate that MobileInst using $T = 3$ can achieve competitive performance, as discussed in Tab. 4. For simpler videos or scenes, the reuse interval T can be further extended for more efficient segmentation and tracking.

Temporal Training via Query Passing

How to fully leverage temporal contextualized information in video for better temporal segmentation is a long-standing research problem in VIS. Whereas, adding additional temporal modules introduces extra parameters and inevitably modifies the current architecture of MobileInst. To leverage temporal information in videos, we present a new temporal training strategy via query passing to enhance the feature representation for temporal inputs, which is inspired by (Yang et al. 2021). Specifically, we randomly sample two frames, e.g., frame t and frame $t + \delta$, from a video sequence during training, as shown in Fig. 4. We adopt the object queries Q_G^t generated from the global instance decoder as the object queries $\tilde{Q}_G^{t+\delta}$ and pass it to the local instance decoder with local features $X_L^{t+\delta}$ to generate $\tilde{M}^{t+\delta}$. The generated $\tilde{M}^{t+\delta}$ shares the same mask targets with $M^{t+\delta}$, and is supervised by the mask losses mentioned in Sec. .

Loss Function

MobileInst outputs N predictions and uses bipartite matching for label assignment (Carion et al. 2020). As the query passing does not require extra module and loss, we follow previous work (Cheng et al. 2022b) and use the same loss function for training MobileInst, which is defined as follows:

$$\mathcal{L} = \lambda_c \cdot \mathcal{L}_{cls} + \lambda_{mask} \cdot \mathcal{L}_{mask} + \lambda_{obj} \cdot \mathcal{L}_{obj}, \quad (2)$$

where \mathcal{L}_{cls} indicates the focal loss for classification, \mathcal{L}_{mask} is the combination of dice loss and pixel-wise binary cross entropy loss for mask prediction, and \mathcal{L}_{obj} indicates the binary cross-entropy loss for IoU-aware objectness. λ_c , λ_{mask} and λ_{obj} are set to 2.0, 2.0 and 1.0 respectively.

Experiments

In this section, we mainly evaluate MobileInst on the challenging COCO (Lin et al. 2014) and Youtube-VIS (Yang, Fan, and Xu 2019) datasets to demonstrate the effects of MobileInst in terms of speed and accuracy. In addition, we conduct extensive ablation studies to reveal the effects of the components in MobileInst. We refer the reader to the arXiv version for additional experiments and visualizations.

Datasets

COCO. COCO dataset is a touchstone for instance segmentation methods, which has 118k, 5k, and 20k images

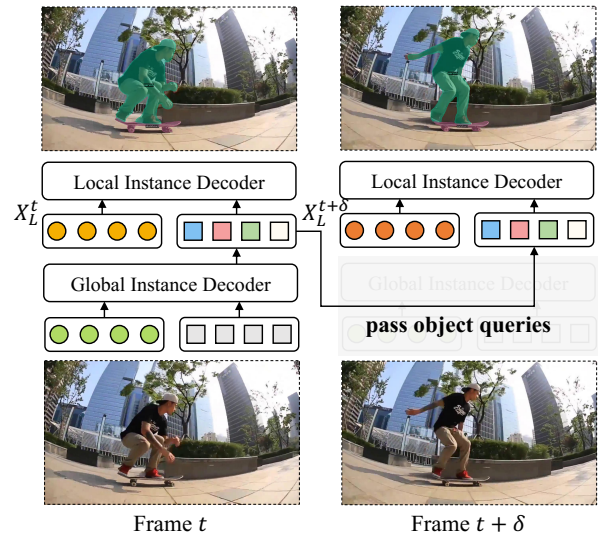


Figure 4: Temporal Training via Query Passing. We sample two frames with an interval δ , e.g., the frame t and the frame $t + \delta$. During temporal training, we adopt the object queries Q_G^t of frame t from the global instance decoder as the object queries $\tilde{Q}_G^{t+\delta}$ and pass it to the local instance decoder with local features $X_L^{t+\delta}$ to generate $\tilde{M}^{t+\delta}$.

for training, validation, and testing respectively. MobileInst is trained on train2017 and evaluated on val2017 or test-dev2017.

YouTube-VIS. YouTube-VIS 2019 is a large-scale dataset for VIS, which has 2,883 videos and 4,883 instances covering 40 categories. YouTube-VIS 2021 expands it to $1.5 \times$ videos and $2 \times$ instances with improved 40 categories. We evaluate our methods on the validation set of both datasets¹.

Implementation Details

Instance Segmentation. We use the AdamW optimizer with an initial learning rate 1×10^{-4} and set the backbone multiplier to 0.5. Following the training schedule and data augmentation as (Cheng et al. 2022b), all models are trained for 270k iterations with a batch size of 64 and decay the learning rate by 10 at 210k and 250k. We apply random flip and scale jitter to augment the training images. More precisely, the shorter edge varies from 416 to 640 pixels, while the longer edge remains under 864 pixels.

Video Instance Segmentation. The models are initialized with weights from the instance segmentation model pre-trained on the COCO train2017. We set the learning rate to 5×10^{-5} and train for 12 epochs with a $10 \times$ decay at the 8-th and 11-th epochs. We only employ basic data augmentation, such as resizing the shorter side of the image to 360, without using any additional data or tricks.

¹ALL Datasets were solely downloaded and evaluated by the University.

method	backbone	size	latency(ms)	AP	AP ₅₀	AP ₇₅	AP _s	AP _M	AP _L
Mask R-CNN (He et al. 2017)	R-50	800	-	37.5	59.3	40.2	21.1	39.6	48.3
CondInst (Tian, Shen, and Chen 2020)	R-50	800	4451	37.8	59.1	40.5	21.0	40.3	48.7
SOLOv2-Lite (Wang et al. 2020c)	R-50	448	1234	34.0	54.0	36.1	10.3	36.3	54.4
YOLACT (Bolya et al. 2019)	R-50	550	1039	28.2	46.6	29.2	9.2	29.3	44.8
SparseInst (Cheng et al. 2022b)	R-50	608	1752	34.7	55.3	36.6	14.3	36.2	50.7
YOLACT (Bolya et al. 2019)	MobileNetV2	550	463	22.2	37.7	22.5	6.0	21.3	35.5
SOLOv2 (Wang et al. 2020c)	MobileNetV2	640	1443	30.5	49.3	32.1	4.2	49.6	67.9
YOLACT (Bolya et al. 2019)	TopFormer	550	497	20.8	37.6	20.2	6.0	20.1	33.5
CondInst (Tian, Shen, and Chen 2020)	TopFormer	640	1418	27.0	44.8	28.0	11.4	27.7	39.0
SparseInst (Cheng et al. 2022b)	TopFormer	608	769	30.0	49.2	30.9	11.0	29.5	46.2
Mask2Former [†] (Cheng et al. 2022a)	TopFormer	640	930	32.0	51.9	33.4	6.9	49.3	68.7
FastInst (He et al. 2023b)	TopFormer	640	965	31.0	50.8	32.0	9.7	31.1	51.7
MobileInst	MobileNetV2	640	410	30.0	49.7	30.8	10.3	30.2	46.0
MobileInst	TopFormer	512	346	29.9	49.4	30.6	9.0	29.2	48.5
MobileInst	TopFormer	640	433	31.2	51.4	32.1	10.4	31.3	49.1
MobileInst	SeaFormer	640	438	31.6	51.8	32.6	10.0	31.5	50.8

Table 1: Instance Segmentation on COCO *test-dev*. Comparisons with state-of-the-art methods for mask AP and inference latency on COCO *test-dev*. The method denoted with [†] was implemented by us.

method	backbone	GPU (ms)	Mobile (ms)	YouTube-VIS 2019				YouTube-VIS 2021			
				AP	AP ₅₀	AP ₇₅	AR ₁	AP	AP ₅₀	AP ₇₅	AR ₁
MT R-CNN (Yang, Fan, and Xu 2019)	R-50	30.1	-	30.3	51.1	32.6	34	28.6	48.9	29.6	26.5
SipMask (Cao et al. 2020)	R-50	29.3	-	33.7	54.1	35.8	35.4	31.7	52.5	34.0	30.8
SGMask (Liu et al. 2021a)	R-50	31.9	-	34.8	56.1	36.8	35.8	-	-	-	-
STMASK (Li et al. 2021)	R-50	28.2	-	33.5	52.1	36.9	31.1	30.6	49.4	32.0	26.4
CrossVIS (Yang et al. 2021)	R-50	25.0	981	34.8	54.6	37.9	34.0	33.3	53.8	37.0	30.1
CrossVIS	TopFormer	24.9	614	32.7	54.3	35.4	34	28.9	50.9	29.0	27.8
SparseInst-VIS [†]	TopFormer	25.4	389	33.3	55.1	34.1	35.3	29.0	50.5	29.2	29.3
MobileInst	TopFormer	22.3	188	35.0	55.2	37.3	38.5	30.1	50.6	30.7	30.1

Table 2: Video Instance Segmentation on Youtube-VIS 2019 and Youtube-VIS 2021. ‘GPU’ denotes NVIDIA 2080 Ti and ‘Mobile’ denotes Snapdragon 778G. The method denoted with [†] was implemented by us.

Inference. The inference of MobileInst is simple. MobileInst can directly output the instance segmentation results for single-frame images without non-maximum suppression (NMS). The inference speeds of all models are measured using TNN framework² on the CPU core of Snapdragon 778G without other methods of acceleration.

Experiments on Instance Segmentation

Firstly, we evaluate the proposed MobileInst on COCO *test-dev* dataset for mobile instance segmentation. As the first instance segmentation model designed specifically for mobile devices, we benchmark our approach against real-time instance segmentation methods. Tab. 1 shows the comparisons between MobileInst and previous approaches.

Among all the methods which use ResNet (He et al. 2016) backbone, Mask R-CNN and CondInst naturally achieve AP above 37. However, the deployment challenges of Mask R-CNN as a two-stage model and CondInst make them less desirable for mobile applications. We observe that MobileInst achieves higher accuracy than the popular real-time approach YOLACT based on R-50, with an increase of 3.4 AP and 600 ms faster speed. Notably, MobileInst obtains faster inference speed and higher accuracy compared to those

methods (Bolya et al. 2019; Wang et al. 2020b; Tian, Shen, and Chen 2020; Cheng et al. 2022b,a) with lightweight backbones (Sandler et al. 2018; Zhang et al. 2022). Tab. 1 shows a remarkable speed improvement of up to 50% compared to the previous state-of-the-art method SparseInst. Compared to the well-established Mask2Former, MobileInst has a similar AP with 100% speed improvement. Fig. 1 illustrates the trade-off curve between speed and accuracy, which further clearly shows the great performance of MobileInst.

Experiments on Video Instance Segmentation

In Tab. 2, we evaluate MobileInst YouTube-VIS 2019 and YouTube-VIS 2021 for video instance segmentation. In terms of latency and accuracy, we mainly compared MobileInst with online methods. As shown In Tab. 2, MobileInst can obtain better accuracy and speed than (Yang et al. 2021; Cheng et al. 2022b) under the same setting. Considering that TopFormer aims for mobile devices and it’s less efficient on GPU. However, it is still evident that MobileInst has superior inference speed on mobile devices.

Ablation Study

Ablation on Instance Decoder. In Tab. 3, We evaluate the performance and speed of different configurations of the instance decoder. Tab. 3 shows that using a single global in-

²TNN: a uniform deep learning inference framework

global	local	AP	AP ₅₀	AP ₇₅	latency	FLOPs
✓		28.7	46.9	29.5	413ms	24.17G
	✓	29.3	48.9	30.2	412ms	24.15G
×2		28.5	46.3	29.5	427ms	24.24G
	×2	29.8	49.3	30.4	426ms	24.22G
✓	✓	29.8	49.4	30.4	427ms	24.24G

Table 3: Ablation on the Instance Decoder (COCO val2017). Both the global decoder and local decoder contribute to improvement. ×2 indicates stacking two decoders. Despite the similar performance, global-local is better than local-local for VIS tasks (refer to Tab. 4).

decoder	w/o tem. training			w/ tem. training		
	T=1	T=3	T=6	T=1	T=3	T=6
global-local	28.8	29.3	28.0	30.1	30.5	29.2
local-local	28.1	28.5	27.8	28.9	29.5	28.6
latency (ms)	184	174	171	184	174	171

Table 4: Ablation on the Query Reuse & Temporal Training (YouTube-VIS 2021). ‘T’ refers to the length of the clip within which we reuse mask kernels of the keyframe. Single-frame clips ($T = 1$) only associate kernels without reuse.

stance decoder or a single local instance decoder leads to a performance drop, which demonstrates the effectiveness of the instance decoder with global features for semantic contexts and local features for spatial details. Stacking two local instance decoders obtains a similar performance with the proposed instance decoder, *i.e.*, 29.8 mask AP. However, Tab. 4 indicates that the *proposed instance decoder with aggregating global contexts is superior to stacking two local decoders in terms of segmenting and tracking in videos.*

In Tab. 5, we mainly focus on the local instance decoder and compare different methods of extracting local features from mask features: no pooling, max pooling with the kernel size of 4 or 8, and average pooling with a kernel size of 8. Although no pooling provides a gain of 0.9 in AP, it also incurs a 50% increase in latency, making it not cost-effective. Additionally, it is worth noting that using max pooling leads to a 0.4 AP gain compared to using average pooling. We believe max pooling naturally provides more desirable local information by filtering out unimportant information, forming a better complementary relationship with the global features used in the global instance decoder.

Kernel Reuse & Temporal Training. We conduct a comparative study of two decoder designs (refer to Tab. 3), *i.e.*, (1) global-local: the combination of a global and a local instance decoder and (2) local-local: two local instance decoders, as shown in Tab. 4. For kernel Reuse, T refers to the length of the clip within which we reuse the mask kernels of the keyframe. Regardless of the model architecture, the reuse mechanism in short-term sequences improves inference speed without performance loss. Compared to the training with only frame-level information, the proposed temporal training brings 1.3 and 0.8 AP improvement for the two designs, respectively. In terms of the global-local and local-local decoders, Tab. 4 shows that global-local achieves bet-

size	pool	AP	AP ₅₀	AP ₇₅	latency	FLOPs
ori.	-	30.7	51.1	31.3	613ms	25.85G
4×4	max	30.0	50.2	30.8	434ms	24.32G
8×8	max	29.8	49.4	30.4	427ms	24.24G
8×8	avg	29.4	48.4	30.3	427ms	24.24G

Table 5: Ablation on the Local Instance Decoder (COCO val2017). The pooling is used to extract local features from the mask features for the local instance decoder. Decreasing the pool size can further improve the accuracy but lower the speed. Notably, max pooling brings 0.4 AP gain.

mask decoder	AP	AP ₅₀	AP ₇₅	latency	FLOPs
SparseInst, 4×conv	30.4	49.6	31.2	524ms	34.69G
SparseInst, 2×conv	29.7	49.2	30.4	445ms	24.11G
SparseInst, 1×conv	29.1	48.8	29.7	405ms	18.82G
FPN, 1×conv	28.7	48.1	29.2	400ms	18.48G
ours	29.8	49.4	30.4	427ms	24.24G
ours w/ SE	30.1	49.9	30.9	433ms	24.37G

Table 6: Ablation on the Semantic-enriched Mask Decoder (COCO val2017). ‘SparseInst’ denotes the FPN-PPM used in (Cheng et al. 2022b).

ter performance on video instance segmentation. Compared to the local-local decoder, the queries (kernels) from the global-local decoder aggregate more global contextual features and benefits more from temporal smoothness in videos, as discussed in Sec. , which is more suitable for videos. Tab. 4 well demonstrates the proposed dual transformer instance decoder for video instance segmentation.

Ablation on the Mask Decoder. Mask features play a crucial role in segmentation quality. Here, we investigate different designs of mask decoders in Tab 6. Compared to FPN with 1× conv, our method achieves 1.1 AP improvement by iteratively utilizing multi-scale information, with a latency overhead of only 6ms. Although stacking convolutions still improves the performance, as seen from the results of SparseInst with 4 stacked 3×3 convs, it leads to a significant burden for mobile devices. The proposed semantic-enhancer (SE) brings a 0.3 AP gain and bridges the gap with less cost.

Conclusion

In this paper, we propose MobileInst, an elaborate-designed video instance segmentation framework for mobile devices. To reduce computation overhead, we propose an efficient query-based dual-transformer instance decoder and a semantic-enhanced mask decoder, with which MobileInst achieves competitive performance and maintains a satisfactory inference speed simultaneously. We also propose an efficient method to extend our MobileInst to video instance segmentation tasks without introducing extra parameters. Experimental results on both COCO and Youtube-VIS datasets demonstrate the superiority of MobileInst in terms of both accuracy and inference speed. We hope our work can facilitate further research on instance-level visual recognition on resource-constrained devices.

Acknowledgments

This work was partially supported by the National Key Research and Development Program of China under Grant 2022YFB4500602, the National Natural Science Foundation of China (No. 62276108), and the University Research Collaboration Project (HUA-474829) from Qualcomm.

References

- Athar, A.; Mahadevan, S.; Osep, A.; Leal-Taixé, L.; and Leibe, B. 2020. STEm-Seg: Spatio-temporal Embeddings for Instance Segmentation in Videos. In *ECCV*.
- Bertasius, G.; and Torresani, L. 2020. Classifying, Segmenting, and Tracking Object Instances in Video with Mask Propagation. In *CVPR*.
- Bolya, D.; Zhou, C.; Xiao, F.; and Lee, Y. J. 2019. YOLACT: Real-Time Instance Segmentation. In *ICCV*.
- Cao, J.; Anwer, R. M.; Cholakkal, H.; Khan, F. S.; Pang, Y.; and Shao, L. 2020. Sipmask: Spatial information preservation for fast image and video instance segmentation. In *ECCV*.
- Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-End Object Detection with Transformers. In *ECCV*.
- Chen, Y.; Dai, X.; Chen, D.; Liu, M.; Dong, X.; Yuan, L.; and Liu, Z. 2021. MobileFormer: Bridging MobileNet and Transformer. In *CVPR*.
- Cheng, B.; Misra, I.; Schwing, A. G.; Kirillov, A.; and Girdhar, R. 2022a. Masked-attention Mask Transformer for Universal Image Segmentation. In *CVPR*.
- Cheng, B.; Schwing, A. G.; and Kirillov, A. 2021. Per-Pixel Classification is Not All You Need for Semantic Segmentation. *arXiv preprint arXiv: 2107.06278*.
- Cheng, T.; Wang, X.; Chen, S.; Zhang, W.; Zhang, Q.; Huang, C.; Zhang, Z.; and Liu, W. 2022b. Sparse Instance Activation for Real-Time Instance Segmentation. In *CVPR*.
- Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Housley, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.
- Fang, J.; Xie, L.; Wang, X.; Zhang, X.; Liu, W.; and Tian, Q. 2022. MSG-Transformer: Exchanging Local Spatial Information by Manipulating Messenger Tokens. In *CVPR*.
- Fang, Y.; Liao, B.; Wang, X.; Fang, J.; Qi, J.; Wu, R.; Niu, J.; and Liu, W. 2021a. You only look at one sequence: Rethinking transformer in vision through object detection. *Advances in Neural Information Processing Systems*, 34: 26183–26197.
- Fang, Y.; Yang, S.; Wang, X.; Li, Y.; Fang, C.; Shan, Y.; Feng, B.; and Liu, W. 2021b. Instances as Queries. In *ICCV*.
- Fu, Y.; Yang, L.; Liu, D.; Huang, T. S.; and Shi, H. 2020. Compfeat: Comprehensive feature aggregation for video instance segmentation. In *AAAI*.
- Han, S. H.; Hwang, S.; Oh, S. W.; Park, Y.; Kim, H.; Kim, M.-J.; and Kim, S. J. 2022. Visolo: Grid-based space-time aggregation for efficient online video instance segmentation. In *CVPR*.
- He, F.; Zhang, H.; Gao, N.; Jia, J.; Shan, Y.; Zhao, X.; and Huang, K. 2023a. InsPro: Propagating Instance Query and Proposal for Online Video Instance Segmentation. *NIPS*.
- He, J.; Li, P.; Geng, Y.; and Xie, X. 2023b. FastInst: A Simple Query-Based Model for Real-Time Instance Segmentation. In *CVPR*.
- He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. B. 2017. Mask R-CNN. In *ICCV*.
- He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.
- Heo, M.; Hwang, S.; Oh, S. W.; Lee, J.-Y.; and Kim, S. J. 2022. Vita: Video instance segmentation via object token association. *NIPS*.
- Huang, D.-A.; Yu, Z.; and Anandkumar, A. 2022. Minvis: A minimal video instance segmentation framework without video-based training. *NIPS*.
- Hwang, S.; Heo, M.; Oh, S. W.; and Kim, S. J. 2021. Video instance segmentation using inter-frame communication transformers. *NIPS*.
- Koner, R.; Hannan, T.; Shit, S.; Sharifzadeh, S.; Schubert, M.; Seidl, T.; and Tresp, V. 2023. InstanceFormer: An Online Video Instance Segmentation Framework. *AAAI*.
- Li, M.; Li, S.; Li, L.; and Zhang, L. 2021. Spatial feature calibration and temporal fusion for effective one-stage video instance segmentation. In *CVPR*.
- Lin, H.; Wu, R.; Liu, S.; Lu, J.; and Jia, J. 2021. Video instance segmentation with a propose-reduce paradigm. In *ICCV*.
- Lin, T.; Dollár, P.; Girshick, R. B.; He, K.; Hariharan, B.; and Belongie, S. J. 2017a. Feature Pyramid Networks for Object Detection. In *CVPR*.
- Lin, T.; Goyal, P.; Girshick, R. B.; He, K.; and Dollár, P. 2017b. Focal Loss for Dense Object Detection. In *ICCV*.
- Lin, T.; Maire, M.; Belongie, S. J.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common Objects in Context. In *ECCV*.
- Liu, D.; Cui, Y.; Tan, W.; and Chen, Y. 2021a. Sg-net: Spatial granularity network for one-stage video instance segmentation. In *CVPR*.
- Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021b. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*.
- Mehta, S.; and Rastegari, M. 2022. MobileViT: Lightweight, General-purpose, and Mobile-friendly Vision Transformer. In *ICLR*.
- Ren, S.; He, K.; Girshick, R. B.; and Sun, J. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.*
- Sandler, M.; Howard, A. G.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*.
- Tian, Z.; Shen, C.; and Chen, H. 2020. Conditional Convolutions for Instance Segmentation. In *ECCV*.

- Tian, Z.; Shen, C.; Chen, H.; and He, T. 2022. FCOS: A Simple and Strong Anchor-Free Object Detector. *IEEE Trans. Pattern Anal. Mach. Intell.*
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NeurIPS*.
- Wan, Q.; Huang, Z.; Lu, J.; Yu, G.; and Zhang, L. 2023. SeaFormer: Squeeze-enhanced Axial Transformer for Mobile Semantic Segmentation. In *ICLR*.
- Wang, W.; Xie, E.; Li, X.; Fan, D.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2021a. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. In *ICCV*.
- Wang, X.; Kong, T.; Shen, C.; Jiang, Y.; and Li, L. 2020a. SOLO: Segmenting Objects by Locations. In *ECCV*.
- Wang, X.; Zhang, R.; Kong, T.; Li, L.; and Shen, C. 2020b. SOLOv2: Dynamic and Fast Instance Segmentation. In *NeurIPS*.
- Wang, X.; Zhang, R.; Kong, T.; Li, L.; and Shen, C. 2020c. Solov2: Dynamic and fast instance segmentation. *NIPS*.
- Wang, Y.; Xu, Z.; Wang, X.; Shen, C.; Cheng, B.; Shen, H.; and Xia, H. 2021b. End-to-end video instance segmentation with transformers. In *CVPR*.
- Wu, J.; Jiang, Y.; Bai, S.; Zhang, W.; and Bai, X. 2022a. Seqformer: Sequential transformer for video instance segmentation. In *ECCV*.
- Wu, J.; Liu, Q.; Jiang, Y.; Bai, S.; Yuille, A.; and Bai, X. 2022b. In defense of online models for video instance segmentation. In *ECCV*.
- Xie, E.; Sun, P.; Song, X.; Wang, W.; Liu, X.; Liang, D.; Shen, C.; and Luo, P. 2020. PolarMask: Single Shot Instance Segmentation With Polar Representation. In *CVPR*.
- Yang, L.; Fan, Y.; and Xu, N. 2019. Video Instance Segmentation. In *ICCV*.
- Yang, S.; Fang, Y.; Wang, X.; Li, Y.; Fang, C.; Shan, Y.; Feng, B.; and Liu, W. 2021. Crossover learning for fast online video instance segmentation. In *ICCV*.
- Yang, S.; Wang, X.; Li, Y.; Fang, Y.; Fang, J.; Liu, W.; Zhao, X.; and Shan, Y. 2022. Temporally efficient vision transformer for video instance segmentation. In *CVPR*.
- Zhang, R.; Tian, Z.; Shen, C.; You, M.; and Yan, Y. 2020. Mask Encoding for Single Shot Instance Segmentation. In *CVPR*.
- Zhang, W.; Huang, Z.; Luo, G.; Chen, T.; Wang, X.; Liu, W.; Yu, G.; and Shen, C. 2022. TopFormer: Token Pyramid Transformer for Mobile Semantic Segmentation. In *CVPR*.
- Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; and Dai, J. 2021. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *ICLR*.