Simple Video Instance Segmentation with ResNet and Transformer

Wenbo Li[1†], Jianzhen Tang[2], Yujia Xie[3]
Huapohen AI Lab
Chengdu Huapohen Technology, China
{liwenbo, tangjianzhen, xieyujia}@huapohen.com

Jian Lan[3]
School of Mechatronic Engineering and Automation
Shanghai University, China
lanjian@shu.edu.cn.

Abstract

Video instance segmentation(VIS) is a new vision task that has emerged in recent years and is processed by deep learning algorithms. It uses continuous video frames as input, generally ranging from a few frames to dozens of frames. Therefore, this segmentation task requires a large batch and large GPU memory for training. Based on recent VIS model VisTR, we propose a competitive VIS model called SimpleVTR in this paper. SimpleVTR trade off and optimizes the computing resources and effects of end-to-end video instance segmentation algorithm. While reducing the computing resources, the model effect can also be well maintained. First, we used one RTX3090 (24G) and one RTX1080Ti (11G) to continuously experiment and optimize the VIS model, and finally determined a model that can train only with one RTX1080Ti (11 G). At the same time, it achieved a score of 31.9 AP on the testing of Video Instance Segmentation in the YouTuBe VOS 2021 Challenge track 2. In the course of the experiment, we summarized some proposals and end-to-end training methods that should be followed.

1. Introduction

Video instance segmentation (VIS) [2][9][10][11] is a new vision task that has emerged in recent years and is processed by deep learning algorithms. VIS occupies an important position in industrial production, such as autonomous driving, film and television post-processing, and video editing. The VIS task uses continuous frames as input. These continuous frames are extracted from a small segment of video, usually ranging from a few frames to dozens of frames, and output continuous frames with the results of the instance mask segmentation. The difference from the classic image instance segmentation task is that the VIS task segments specific instances from consecutive frames, and at the same time, the instances between frames can be matched. However, the classic segmentation algorithms such as Mask R-CNN[17] and SOLO[6] do instance segmentation for a single image, and the instances between images within a batch do not need a matching relationship. The recent processing methods of VIS tasks such as MaskTrackRCNN[9], STEm-Seg[11], etc. require carefully designed modules to process images and then perform tracking and matching, leaving the question of speed and complexity. The newly proposed method VisTR[2] is based on DETR[1]. It processes continuous frame images end-to-end without complicated post-processing. The VisTR model architecture is simple and the effect is competitive, but there are problems such as large memory usage of the model, not friendly to small objects, and insufficient fineness of the instance mask. Based on VisTR, we started from model optimization to solve the problem of GPU memory consumption, and finally used a GTX1080Ti for training. In order to use Transformer[3] for VIS tasks, some optimization proposals are pointed out.

We empirically pointed out the following proposals:

- DCN[12] is an important component in VisTR. Because the feature map of the instances is obtained by copying feature map in mask head FPN[7], the training difficulty increases after removing DCN and DCNv2[13] is more adaptable to network adjustments than DCN.
- The mask head can be simplified and some modules such as 3D convolution can be removed, but it needs to follow the processing pipeline of DETR's FPN.
- The number of layers of Transformer's encoder and decoder can be reduced, for example, from 6 to 4.
- In our experiment, using the AdaBelief[15] optimizer can speed up training, and the learning rate setting is not greater than 1e-4.
- During training and testing, choosing different random seed has an impact of 1 point on AP.
- The way the data set is processed has a great influence on training.
- After decoupling the frame and the instance in the query embedding, any frame can be inferred. The training uses 16 frames and the inference is 64 frames

We finally submitted an optimized version to the testing server of the 3rd Large-scale VOS 2021 Challenge Track 2, called SimpleVTR, which contains the above proposals.

2. Related Work

Mask Track R-CNN. Based on Mask R-CNN[17], the VIS framework MaskTrackRCNN[9] extends from image instance segmentation to video instance segmentation. Calculate the instance match among frames by adding a tracking header. The model uses the features extracted by the backbone as the memory, and the tracking module extracts the features from the memory. In this method of segmentation, the instances between frames are processed separately, and only on the tracking head, additional matching is performed. Multi-step processing will make the processing speed slow, and the information of the same instance between frames is not used for segmentation.

DETR. DETR[1] is based on ResNet[16] framework and Transformer[3] framework. As backbone, ResNet is used as the CNN feature extractor to extract the features of the last layer of each block. The feature of the output layer of the last block are used as the memory of the Transformer[3]. DETR uses the seq2seq[18] mechanism of the encoder-decoder of the Transformer framework to perform sequence prediction. The prediction of the network output uses the Hungarian algorithm to construct a unique prediction through bipartite graph matching. The instance segmentation head of DETR is a simplified FPN[7]. DETR is different from the classic single-stage and two-stage target detection algorithms, and there is no elaborate manual design such as NMS post-processing and predefined anchor.

VisTR. On the basis of DETR, VisTR[2] extends the original processing pipeline of single-frame images to the processing pipeline of consecutive frames for VIS. In the feature extraction stage of ResNet, the continuous frame images are placed in the batch channel. In the Transformer stage, the batch channel size is set to 1, and the number of consecutive frames and instances are combined in one channel at the same time, and the size is equal to the number of frames multiplied by the number of instances. VisTR is based on the sequence processing capabilities of the Transformer in the DETR framework, and processes the instance tracking among consecutive frames through the Hungarian algorithm. VisTR uses additional defined query embedding to query the cross-frame instance in the output feature layer of the backbone, and processes consecutive frames at the same time, achieving a balance of speed and efficiency, and constructs an end-to-end processing method to make real-time video instance segmentation possible. However, in the model inference stage, the fixed frame input limits the number of video frames, and for a video with a small number of frames, it will take the same time as a fixed frame in the depth model stage. In addition, the example mask output is zoomed from bilinear interpolate resolution close to 1/12 of the original image to the original image resolution, which is not friendly to small obejct instances. Most important, in the training phase, model requires a GPU with large memory, such as V100(32G).

3. SimpleVTR

In this paper, SimpleVTR follow the design patterns of DETR and VisTR, and move towards optimizing the VisTR network first to reduce the computing resources and stabilize the index effect, and then explore more network architectures to improve the effect. Our experiments is trained on VOS 2021 Training Dataset and is tested on the development server of VOS 2021 Challenge Track 2. Finally, the version with the smallest changes we called SimpleVTR was submitted to the testing server of VOS 2021 Challenge Track 2. The main contribution is: We have pointed out which optimization directions are feasible, and we will introduce these changes in detail next.

3.1. Architecture

The architecture of SimpleVTR is shown in Figure 1. Without bells and whistles, we aim to reduce computing resources with minimal changes, and specially marked the key operation DCN for the VIS task based on VisTR.

3.2. Deformable Convolution

We want to explore what is the indispensable module added to the VisTR architecture on the basis of DETR. We located the 3x3 DCN[12] module. Considering that 3x3 DCN has higher complexity and memory usage than ordinary 3x3 convolution, we first replace the 3x3 DCN module with ordinary convolution in FPN. During the experiment, we observed some interesting phenomena. Without the DCN module, it is difficult to reduce the loss during training. In addition, after training a network with a DCN module for a period of time, and then replacing it with an ordinary convolution module, the loss will rise sharply. It can be seen that the DCN module is very important to the network.

The reason may be that the FPN processing of the mask head uses a copy method to artificially create an instance channel, and the channel size is the number of predefined instances. The instances slides smoothly between consecutive frames, and the GIOU[19] of two adjacent frames in time is large. DCN is just right to deal with the difference of instance masks between consecutive frames, while the convolutional field of view of ordinary convolution is limited. At this time, the ordinary convolutional layer stack is small, and it is not enough to deal with the consecutive frame instances obtained by copying like DCN processing. VisTR[2] did not make any relevant explanations about the DCN phenomenon.

The DCN version used by VisTR can run on GTX 1080 Ti, but it is not compatible with the latest graphics card RTX 3090 because of its high computing power. For this reason, we installed DCN v2[13] version on RTX3090. In the FPN of the mask head, as the resolution gradually increases, the memory occupation of the feature map also gra-

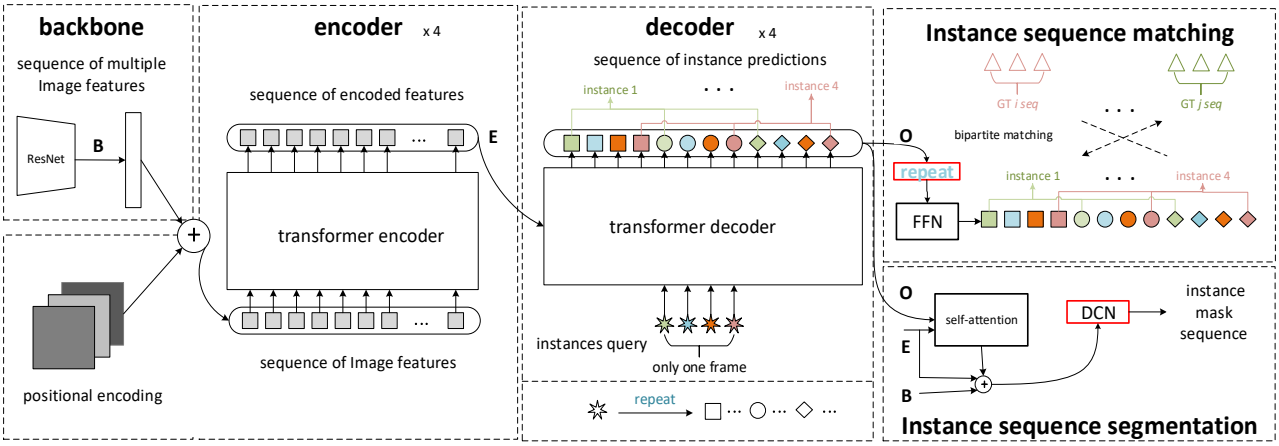

Figure 1: Architecture of SimpleVTR. We decouple *frames* and *instances* to reduce computing resources, and restore *frames* through *repeat* operation. Key operation DCN is marked with a red box. In the inference stage of SimpleVTR, *frames* can be set to any size.

dually increases. The last few layers close to the output layer are the most occupied areas of the network. The version of DCN v2 we installed separately occupies the GPU memory for different resolution feature maps. Because the continuous frame size of the network input has been resized within a predetermined range, DCN v2 will quickly fill up the GPU memory and cause an error. This limits the use of the DETR-based VisTR framework. We explored alternatives to replace DCN. Because the alternatives were not submitted to the VOS competition testing server, we only point out here: If there is no architectural change to the FPN of the VisTR mask head, then the added DCN module will be a big trick and an indispensable key operation for optimizing the VisTR-like network. SimpleVTR retains the original DCN to experiment on GTX1080Ti, and we have stacked two layers of 3x3 DCN at the output layer.

3.3. Backbone

The backbone of DETR uses ResNet [16], which is only used for feature extraction. As the number of frames increases, the GPU memory occupied by ResNet will increase accordingly. Both VisTR and DETR have not explored the redundancy of the backbone in Transformer-based tasks, and we want to save the small goals ignored by the VisTR architecture, because classification and bbox only use the last layer of the 4 layers taken from the backbone. We use Transormer[3][29] or improved Tansformer vision architecture[25][26][27][28] as backbone to replace ResNet, and integrate VisTR's Transformer module into backbone. The classification and detection task can be effective but the VIS can not. In the end, in the SimpleVTR version submitted to the 2021 VOS testing server, we still retained the original ResNet as backbone.

3.4. Query Embedding

The query embedding of SimpleVTR is defined as:

$$self.query\_embed = nn.Embedding(self.instances, hidden\_dim) \quad (1)$$

In the above formula, *self.instances* is the predefined maximu-m number of instances of a single frame *10*. There are no frames in the definition of query_embed. We separate frames and instances, keep instances and ignore frames. The decoupling of frames and instances is a key step in the single-card training VIS task. While reducing the computing resources, it also facilitates the input of arbitrary frames for the inference stage. We consider that the instances will almost always exist in consecutive frames, except for a few occlusion situations, which is OVIS[10] task. After *self.query_embed* is processed by Transformer, we will take a simple copy operation to reflect the frames。

3.5. Transformer

Transformer is derived from Attention Is All You Need[3] and has a powerful ability to process sequences. We take the features of the last layer taken out by backbone as memory for Transformer to query, that is, output the feature maps required by class task and bbox task. We reduce the encoder and decoder layers of Transformer from $k=6$ to $k=4$. Because query embedding decouples frames and instances, it is faster to operate and consumes less GPU memory. The query embedding output by Transformer needs to be copied in $N=frames$ copies, written as follows with einops[21] expression:

$$x = repeat(x, \text{"k 1 ins hw} \rightarrow \text{k 1 (frames ins) hw"}, \text{frames} = \text{self.frames}) \quad (2)$$

The output feature ***x*** is used as input for *class_head*, *bbox_head*, and *mask_head*. SimpleVTR follows the processing framework of VisTR, and the effect of copying *N=frames* ***x*** on *class_head* and *bbox_head* is acceptable according to the experimental results. The reasons for this analysis may be:

- the category of the instance does not change among consecutive frames
- *bbox_head* does not participate in the mask calculation, nor does it participate in the inference of the video instance segmentation. For the segmentation task, only the loss of bbox is used when calculating the total loss.

3.6. Mask Head

In SimpleVTR, we removed the 3D convolution derived from VisTR, the output layer is stacked with two DCNs, the kernel size of the convolution is 3, and the second DCN directly outputs the instance segmentation mask. We have observed that after the feature map is gradually enlarged in resolution through FPN, the memory usage will gradually increase, especially after FPN performs the last nearest neighbor interpolation. In order to adapt to the GPU memory of the GTX 1080Ti single card, we directly cancelled the 3D convolution, because we considered that after the copy operation, a new *instance* channel is added, and the 3D convolution is based on DCN to further integrate features, and its contribution to AP is limited. In other words, the more important operation is DCN instead of 3D convolution.

The FPN of *mask_head* enlarges the resolution of the feature map to close to 1/12 of the resolution of the original image, which is extremely unfriendly to small object instances. It is necessary to use GPU memory saved by the optimization to enhance small objects segmentation. We tried some improvement measures[23][24]. Because we did not submit these measures to the VOS 2021 testing server, we only pointed out that the *mask_head* module in SimpleVTR only achieves the effect of preserving the core module DCN and reducing computing resources, but does not improve the effect of the small object VIS.

4. Experiments

We first analyzed the training dataset and development data set of VOS 2019 and VOS 2021, so that in the process of optimizing VisTR to obtain SimpleVTR, there are reference standards for setting the number of frames and the number of instances. First, we pointed out the possible over fitting problem when performing video instance segmentation based on VisTR , which must start with the data set. Second, we tried to adjust the learning rate and the choice of the optimizer, solving the problems caused by the resizing of the input image. Thrid, we pointed out the direction of the optimization we have tried. Finally, we submitted the SimpleVTR to the testing dataset, trained with a GTX1080Ti, and got 31.9AP.

4.1. Dateset

We analyzed the datasets[9] of VOS 2019 and VOS 2021. In the VOS2019, the maximum number of single-frame instances is *6*, which is set to *10* in the VisTR. It means that *40%* of the GPU memory resources are wasted, in the mask head step that occupies the most GPU memory. In the VOS 2021 , the maximum number of single frame instances is *25*, which we preset to *10* in SimpleVTR. The selection of the frames number will also affect the occupancy of the GPU memory and the final training effect. We put the analysis results of the instances and the frames number of in the appendix. The preprocessing methods of dataset is according to cocoapi[22].

When processing the VOS2021 training dataset, we have designed two schemes. First, set the input frame size as *N=frames*.

- The scheme one is taking the video as the unit of input data, which means inputting the consecutive frames of the entire video. So the size of dataset is *2985*, which is the total number of videos.
- The scheme two is taking the frame as the unit of input data, which means inputting all frames of all videos. So the size of dataset is *90160*. In each iteration, each frame needs to be processed separately into the fixed *N=frames*, which is filled in with the frames of the video where it is located.

The scheme one treats all videos equally, and reuses video frames to fill in when frames number of the video is short. Each epoch will randomly shuffle consecutive frames in a video. The training time of one epoch is short. The scheme two is biased towards videos with a large number of frames or with a large number of categories. So it is not friendly to the videos which have a few frames or a few number of videos for a special category, which will cause the trained model towards the head categories, and even overfit. We use VisTR's pre-training parameters on the VOS 2019 training dataset to inference the VOS 2021 development dataset, and get a 0.001 AP. The reason why the AP is only 0.001 is related to the processing methods of the dataset. VisTR has made data enhancements to the VOS 2019 dataset. Based on the preprocessing of the DETR dataset, VisTR adds brightness, saturation , Hue, Contrast, HSV, noise and other methods to enhance the dataset, increase diversity, and also ease overfitting.

We tried scheme one, scheme two, and a compromise between scheme one and two. The compromise solution can achieve better results, but the SimpleVTR that we submitted to the test dataset of the challenge adopts the second solution.

4.2. Model Settings

The SimpleVTR we proposed inherits VisTR's hyperparameter settings based on DETR. The seed is set to 666 during inference, because we observe that the seed setting is different and mAP fluctuates close to 1 point. In the experiment, we found that when the learning rate is set to *1e-3*, the loss cannot be reduced from the scratch training. We set the learning rate to *1e-4*, and after *5* epochs, the learning rate drops to *1e-5*. For the optimizer, we did not use AdamW[14] or Adam[20] but chose AdaBelief[15], in whichh *weight_decay* is set to *1e-4*, *weight_decouple* and *rectify* are both set to *True*. The number of encoder and decoder layers of Transformer is set to *4*. The number of input frames during training is fixed at *16*. Due to *frames* and *instances* are decoupled by query embedding, we tried different sizes of input frames during inference, and found that AP was the highest when the number of input frames was set to the number of training frames. Finally, on the testing dataset, the number of inference frames we set was *64* frames. We found an interesting phenomenon. SimpleVTR inherits VisTR's inference pipline and sets different frame numbers when inferring on the development dataset, and the obtained APs are not much different.

4.3. Image size and Resolution

Before adopting the SimpleVTR version, we tried to fix the input image resolution to 360 or 640. As a comparison, VisTR adopted the resolution to 300. Most of the video resolution in the VOS dataset are 1280x960, and a few are 640x480, 1920x1080, and other resolutions. Take 1280x960 as a reference, because VisTR will resize the input resolution to 300 during data preprocessing. For the convenience of discussion, suppose the output resolution of VisTR is 100x75, which is $\frac{1}{12.8}$ of the original resolution 1280x960. During training, when the input or output resolution is increased, the GPU memory occupied by query embedding in the Transformer stage will become very large, and the GPU memory occupied by the *mask_head* stage will also become very large. Based on the VisTR framework, one of the shortcomings of SimpleVTR is that the network output resolution is very small, which leads to the loss of small object instances after 100x75 enlarged to 1280x960 through bilinear interpolation during inference. We tried some solutions to increase the resolution, but did not achieve good results. Because the VIS task has both the *frame* channel and the *instance* channel, the resolution of the feature map cannot be enlarged under the DETR framework. The primary goal of SimpleVTR is not to improve performance, but to reduce computing resources. The version we submitted to the testing dataset, SimpleVTR, directly reuses VisTR's FPN processing, and does not include the processing of resolution enlargement.

4.4. Results

The effect of SimpleVTR on the test dateset of 2021 VOS challenge track 2 video instance segmentation is shown in the following tables.

AP	AP_{50}	AP_{75}	AP_{small}	AP_{medium}	AP_{large}
31.9	51.5	33.8	9.2	38.6	53.1

Table 1: Average Precision of SimpleVTR

AR_1	AR_{10}	AR_{small}	AR_{medium}	AR_{large}
31.2	37.9	11.3	42.9	59.2

Table 2: Average Recall of SimpleVTR

SimpleVTR achieved 31.9 mAP. In addition, the AP of the small object is very small, and the AP of the large object and the small object are quite different. At the same time, the AR of small object is also very small。

5. Conclusion

SimpleVTR optimizes the network based on the structure of VisTR, which reduces the computing resources required during training, and brings the possibility of training a larger number of video frames and a larger number of instances. Although SimpleVTR did not improve the effect of instance segmentation, it pointed out some problems and key points in the original VisTR, and pointed out a feasible direction for improving the segmentation effect in the future.

Reference

[1] NicolasCarion, FranciscoMassa, GabrielSynnaeve, Nicolas Usunier, AlexanderKirillov, and SergeyZagoruyko. End-to-End Object Detection with Transformers. In *ECCV*, 2020.

[2] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, Huaxia Xia. End-to-End Video Instance Segmentation with Transformers. In *CVPR* 2021.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NeurIPS*, 2017.

[4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[6] Xinlong Wang,Tao Kong,Chunhua Shen,Yuning Jiang,and Lei Li. SOLO: Segmenting objects by locations. In *ECCV*, 2020.

[7] Tsung-Yi Lin, Piotr Doll´ar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, 2017.

[8] Tsung-YiLin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal Loss for Dense Object Detection. In *ICCV*, 2017.

[9] Linjie Yang, Yuchen Fan, and Ning Xu. Video Instance Segmentation. In *ICCV*, 2019.

[10] Jiyang Qi, Yan Gao, Yao Hu, Xinggang Wang, Xiaoyu Liu, Xiang Bai, Serge Belongie, Alan Yuille, Philip H.S. Torr, Song Bai. Occluded Video Instance Segmentation. arXiv preprint arXiv:2102.01558, 2021

[11] Ali Athar, Sabarinath Mahadevan, Aljoˇsa Oˇsep, Laura LealTaix´e, and Bastian Leibe. Stem-seg: Spatio-temporal embeddings for Instance Segmentation in Videos. In *ECCV*, 2020.

[12] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable Convolutional Networks. In *ICCV*, 2017.

[13] Xizhou Zhu, Han Hu, Stephen Lin, Jifeng Dai. Deformable ConvNets v2: More Deformable, Better Results. In *CVPR*, 2019.

[14] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2017.

[15] Juntang Zhuang, Tommy Tang, Sekhar Tatikonda, Nicha Dvornek, et al. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. In *NeurIPS*, 2020.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[18] Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *NIPS*, 2014.

[19] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, et al. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. In *CVPR*, 2019.

[20] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2014.

[21] Alex Rogozhnikov, Cristian Garcia, et al. Einops, https://github.com/arogozhnikov/einops, 2018

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.

[23] Jonathan Long*, Evan Shelhamer*, Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*, 2015.

[24] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, et al. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *CVPR*, 2016.

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. arXiv preprint arXiv: 2103.14030, 2021.

[26] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, Chunhua Shen. Twins: Revisiting the Design of Spatial Attention in Vision Transformers. arXiv preprint arXiv: 2104.13840, 2021.

[27] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, et al. MLP-Mixer: An all-MLP Architecture for Vision. arXiv preprint arXiv:2105.01601, 2021.

[28] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, et al. ResMLP: Feedforward networks for image classification with data-efficient training. arXiv preprint arXiv:2105.03404, 2021.

[29] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021.

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

A Appendix

A.1 2019 VOS Training Dataset analysis

ins	vid	videos percent	cumulative percent1	ins*vid	ins*vid percent	ins*vid cumulative	cumulative percent2
1	1105	49.37%	49.37%	1105	29.28%	1105	29.28%
2	835	37.31%	86.68%	1670	44.25%	2775	73.53%
3	209	9.34%	96.02%	627	16.61%	3402	90.14%
4	77	3.44%	99.46%	308	8.16%	3710	98.30%
5	8	0.36%	99.82%	40	1.06%	3750	99.36%
6	4	0.18%	100%	24	0.64%	3774	100%

Table 3: Instances of 2019 VOS Training Dataset. In table 3, *ins* represents the number of instances, and *vid* represents the number of videos. The maximum number of training set instances in 2019 VOS training dataset is *6*.

A.2 2021 VOS Training Dataset analysis

ins	vid	videos percent	cumulative percent1	ins*vid	ins*vid percent	ins*vid cumulative	cumulative percent2
1	1304	43.69%	43.69%	1304	20.75%	1304	20.75%
2	1008	33.77%	77.46%	2016	32.09%	3320	52.84%
3	328	10.99%	88.45%	984	15.66%	4304	68.50%
4~6	253	8.48%	96.92%	1136	18.08%	5440	86.58%
10	36	1.21%	98.13%	360	5.73%	5800	92.31%
7~25	56	1.88%	100%	483	7.69%	6283	100%

Table 4: Instances of 2021 VOS Training Dataset. In table 4, *ins* represents the number of instances, *vid* represents the number of videos, and *7~25* means all the frames from *7* to *25* are merged together. The maximum number of training set instances in 2021 VOS training dataset is *25*. The number of instances predefined by SimpleVTR is *10*, including *98%* of videos and *92%* of instances.

frame	vid	vid percent	cum_vid	cum_vid percent	frame*vid	frame*vid percent	cum_frame	cum_frame percent
72	1	0.03%	1	0.03%	72	0.08%	72	0.08%
62	1	0.03%	2	0.07%	62	0.07%	134	0.15%
37~60	317	10.62%	319	10.69%	15510	17.20%	15644	17.35%
33~36	1009	33.80%	1328	44.49%	36212	40.16%	51856	57.52%
17~32	1554	52.06%	2882	96.55%	37024	41.06%	88880	98.58%
9~16	97	3.25%	2979	99.80%	1240	1.38%	90120	99.96%
5~8	6	0.20%	2985	100%	40	0.04%	90160	100%

Table 5: Frames of 2021 VOS Training Dataset. In table 5, *frames* represents the number of video frames, *vid* represents the number of videos, *cum_vid* represents the number of accumulated videos, and *cum_frame* represents the number of accumulated frames. The number of video frames predefined by SimpleVTR is *16*, and the proportion of videos with more than *16* frames is *96.55%*. For a video larger than *16* frames, such as *60* frames, *16* frames are randomly selected from this video during training. Because the shape and position of an instance change slowly in few short continuous frames, just like the SOLO[6] instance segmentation framework defines an instance. In addition, because there are only two videos with more than *60* frames, and the median number of frames is *30~31*, *16* frames can represent half of the continuous frames of most

videos. In the data preprocessing stage of each iteration, we randomize the continuous frames in a video, which can be considered that the frame interval of frame sampling becomes larger. Therefore, we assume that *16* frames can represent this video for video instance segmentation. The experimental results show that *16* frames have also been trained to achieve good results, which greatly reduces the GPU memory consumption.

We also analyzed the 2021 VOS Valid Dataset, and the analysis results of the valid dataset are basically consistent with the analysis results of the training data set.

A.3 2021 and 2019 VOS Training Dataset Categories

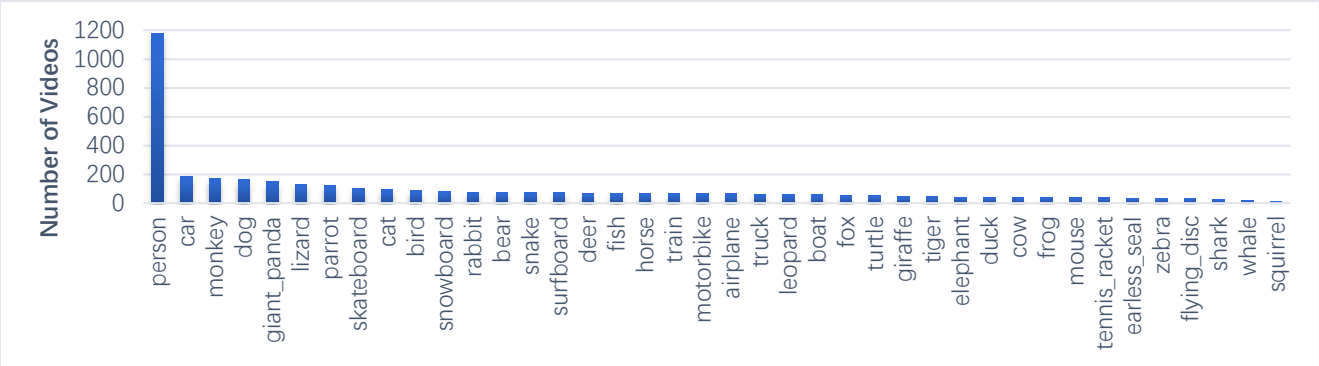

Figure 2. Number of videos per category in the 2021 VOS Training Dataset.

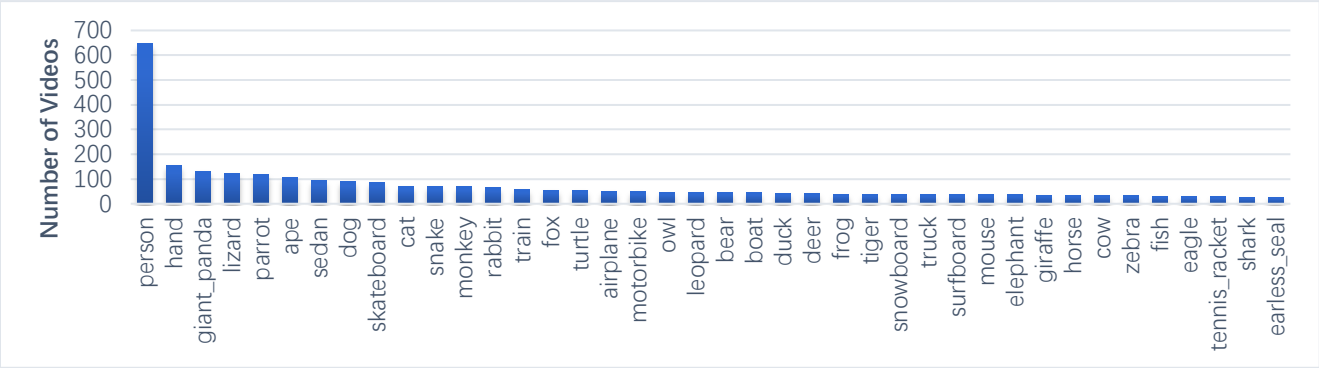

Figure 3. Number of videos per category in the 2019 VOS Training Dataset.

Figure 2 and Figure 3 list the statistical numbers of instance categories of VOS Training Dataset 2021 and 2019, respectively. It can be seen that the number of instance videos of the category *person* is far more than that of other categories. Even though methods such as Focal Loss[17] and DICE[30] are used, the network trained by this approach of VisTR still cannot be well generalized from the 2019 VOS Dataset to the 2021 VOS Dataset. We tried other methods to alleviate over-fitting, but the SimpleVTR version we submitted to the challenge testing server did not include these methods.