Tracking Algorithm using Sequentially Updated Features for Video Instance Segmentation

Software Capstone Design, Yonsei University
Fall, 2020

Janghyuk Choi Dongeun Lee Jinwoo Kim

Supervised by Prof. Seonjoo Kim,
Computational Intelligence and Photography Lab, Yonsei University

1. Introduction

Computer vision field has achieved huge development thanks to the deep learning algorithms and high-performance GPUs. There are lots of computer vision tasks like classification, detection, segmentation, and etc. Among them, segmentation is one of the fundamental problems. Furthermore, not only the image tasks but also the video tasks have become more important to deal with the real world problems. One of the tasks dealing with both segmentation and video is the **Video Instance Segmentation[1]**, which contains simultaneous detection, segmentation and tracking of object instances in videos.

To handle the VIS task, we take the baseline model as MaskTrackRCNN introduced with the VIS task in [1]. MaskTrackRCNN is a network that adds tracking branch to MaskRCNN[2], which is designed for existing instance segmentation. In this project, we aim at **performance enhancement in the VIS task by improving the tracking branch of MaskTrackRCNN** and suggest the model **SUF, Sequentially Updating Feature maps**, which solve the limitation of the baseline model.

Contribution: Our model's contributions are summarized as follows. (i) Our model is trained in the similar way with the inference in stochastic settings. (ii) In inference, our model uses the whole previous instances features to capture the video-level features of each instance in online way.

2. Related Works

2.1. Image segmentation

Segmentation in computer vision is the task dealing with the pixel-level classification. By pixel-level classification, we can obtain masks which tell us the area of the instances. There are two main types of segmentation: semantic segmentation, and instance segmentation. In semantic segmentation, the goal of the task is to classify the category of each pixel, which means that some instances with the same category will be contained in one mask. In instance segmentation, the task aims at classifying pixel-level instance classification, which classify every instance even with the same category.

MaskRCNN is one of the famous networks for dealing with the segmentation task. By adding a 1x1 convolution layer to FasterRCNN[3], MaskRCNN predicts the pixel-level instance ids to make the masks for images. In this project, we starts from the MaskTrackRCNN which is the video version of the MaskRCNN.

2.2. Video Instance Segmentation

There are some similar tasks like MOT(Multiple Object Tracking) whose goal is to track multiple objects without segmentation (or even detection), and VOS(Video Object Segmentation) which is very similar to VIS but dealing only with one object whose information in the first frame is given. VIS is the task of mixing MOT and VOS.

The video instance segmentation(VIS) task requires classifying, detecting, segmenting, and tracking in videos simultaneously. In this task, each video has some instances with variable categories; the number of categories is 40. Each instance should be classified as its category, detected with a bounding box, segmented with a mask, and also tracked by its object id throughout the video.

2.3. MaskTrackRCNN

MaskTrackRCNN is suggested as a baseline model for the VIS task in [1]. By adding tracking branch to the MaskRCNN [2], MaskTrackRCNN can handle video tasks. For VIS task, videos are given to the model as inputs, and the model outputs are mask tracks of instances in the videos with their classes, masks, and object ids.

In tracking branch, the model has the memory queue to store the previous instances' feature map. Then, the model calculates the matching score with current instances in the frame of the video and previous instances in the memory queue with their visual similarity by inner product and convolutional layer to match the ids of instances. And we call it online model since it uses only previous and current frames, not future frames in inference step. In this project, we found some limitations of this tracking branch, and improved the tracking branch to enhance the performance of the model on the VIS task.

2.4 LSTM

Long-Short Term Memory(LSTM) [4] is one of the recurrent neural network(RNN) used in the field of deep learning. LSTM can deal with sequential input data in recursive way. Because of its characteristic, we think it is well suited for treating video-level tracking task.

In this project, we replace the previously naive matching algorithm which only compare a query object with latest instances with our method SUF which utilizes LSTM to capture the video-level instance features.

3. Data

This project uses Youtube-VOS dataset and it has been released at https://youtube-vos.org/home. Youtube-VOS dataset split into train, valid, and test videos with various number of youtube videos with diverse categories. The number of train videos is 3,472, valid is 507, and the number of test videos is 541.

Train and valid videos have annotation information. We use train videos as train data, and check performance by validation videos via competition server not test videos. Unfortunately, because of the COVID-19, competition has not held this year and it's impossible to check performance by test videos.

In this dataset, Annotations of training data provides not only the size of each frame, but also the category ID of the object, the mask labeling result for the object, and information about the bounding box and width.

Each object's class ID consists of 40 categorites, such as 'person', 'giant_panda', 'lizard', 'parrot', etc.

In the case of a bounding box, the first two elements represent (x,y) values of the upper-left coordinates. Last two elements are each width and height. The provided mask coordinates are encoded in RLE format.

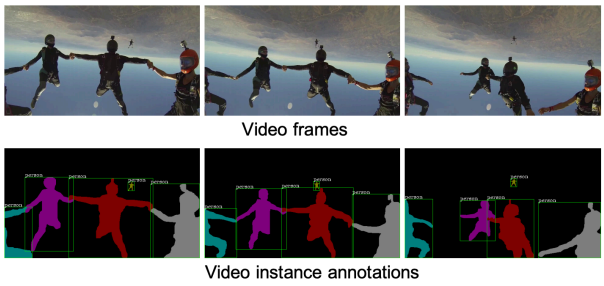

Figure 1. A illustration of video instance segmentation

4. Method proposals

4.1. Sequentially Updating Feature-maps (SUF)

4.1.1. Lack of information in original memory queue

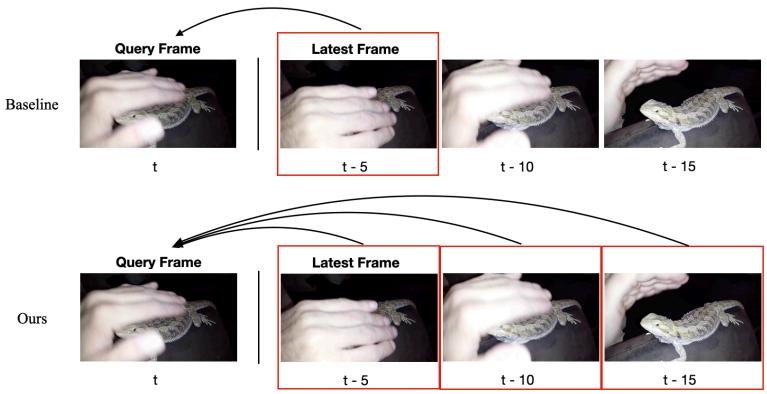

Figure 2. For the query frame we use not only the latest frame, but also the representative frame

MaskTrackRCNN stores the features of previously identified instances into the external memory. The external memory is dynamically updated when a new candidate box is assigned with an instance label so that the external memory keeps track of the latest state of the instance. However, sometimes the latest state of the instance does not give enough information to track the instances. Even more, it might confuse the external memory with less informative features. Figure 2 shows a case where the latest state of the instance is less informative to track an instance than states few frames ago. In that case, if the tracking branch can refer to more features from the past video frames in addition to the latest frame, it will be able to complement the lack of feature information in the latest state. Therefore, we intended to improve the tracking performance by modifying the external memory to make the feature maps keep the information from the previous states not only from the latest state. Since LSTM can accumulate the information through the previous states in sequential manner, we adopted LSTM to enhance the tracking performance by sequentially updating feature maps for each instance. Even though we apply LSTM to tracking branch, since the model only refers to the previous video frames, it still works in online way.

4.2.2 Utilizing LSTM to Sequentially Update Feature maps

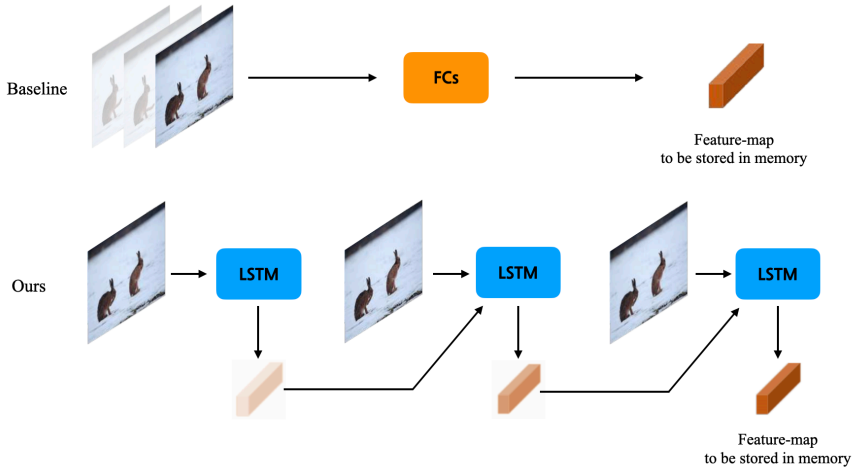

Figure 3. Original memory queue only keeps the feature map of the latest state for each instance. However, with LSTM, our new memory queue will be able to keep the sequentially updated feature map through the previous video frames.

Our strategy to apply LSTM to the tracking branch is as follows. We directly used the LSTM's hidden states as the feature maps for each tracking instance in the external memory, and the feature maps in the memory are updated whenever a new candidate box is assigned to the corresponding instance label. For memory update, the feature map for the candidate box in the current frame is fed as input vector to LSTM, and the feature map for the instance in the external memory is fed as a hidden state to LSTM. By sequentially updating the external memory in this manner, the feature maps are able to keep the accumulated information through the video in addition to the latest state of the instance.

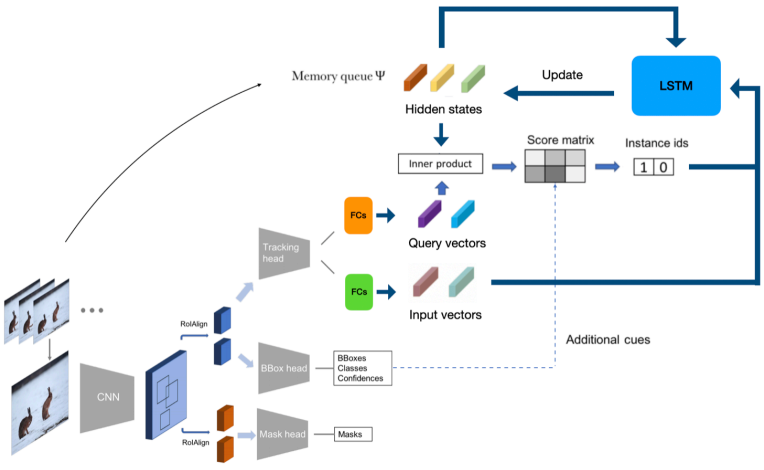

Figure 4. The flowchart of our proposed model SUF.

As shown in *Figure 4*, the matching algorithm is same to that of original tracking branch, where the dot product between the feature map in the memory and the feature map of candidate box corresponds to the match score which is combined with the additional cues. However, the thing to note in our method is that we distinguished the feature map for the calculation of matching score,

denoted as query vector, and the feature map for the LSTM's input vector, denoted as input vector. As the feature maps for candidate boxes initially have the shape of 256x7x7, they are fed to the FCs so that the feature maps are converted to the vector which has the size of 1024. Therefore, to distinguish query vectors and input vectors, we introduced two separated FCs modules instead of only using a single module of FCs.

4.2.3. How to train LSTM

In previous baseline model, MaskTrackRCNN, pairs of images, one image for query and one randomly selected image for reference are used to train the tracking branch. Since the memory queue in inference save the latest feature of each instance, this method is the only way to train the tracking branch in stochastic way.

However, we thought this training mechanism is not suitable to fully learn the instances' tracking algorithm since the way of training and inferencing is too different. To push this envelop, we tried to make the train procedure as similar as inference procedure. In our model SUF, each step of training is done not only randomly selected a pair of images, but with a sequence of images consisting of one query image, and a sequence of reference images. This sequence of reference images is given to LSTM to finally make one feature map of the instance and calculate the matching score with the object in the query image.

Our memory update and matching score calculation in training mechanism is same as the ones in inferencing procedure so that the learning can be done effectively. To give some randomness to prevent overfitting and give some variation, we randomly pick 2~8 images from a whole instance sequence with maximum time gap 3. Since playback direction(forward/backward) doesn't have the absolute information, which means that videos played backwards are still valid, we choose sequences of reference images from previous or next part of the query frame images.

4.2.3. Experiments

	MaskTrackRCNN	*LSTM ver.1*	*LSTM ver.2*	*SUF*
AP on validation set	30.3	32.0	33.0	**33.9**

Table 1. Experiments result

Table 1 shows APs for baseline, *MaskTrackRCNN* and our intermediate model with LSTM. *LSTM ver.1* is the method where LSTM is simply applied to the tracking branch. Unlike SUF where the dot product is used, in Naive LSTM, matching score is derived by feeding LSTM's hidden states to the FCs instead of the dot product. In other words, when computing the matching score against the instances in the external memory, the feature map for each instance in the external memory is fed as hidden state to LSTM individually for a single input vector which is the feature for a candidate box.

LSTM ver.2 is same to our proposed method except that the query vector and input vector is not distinguished. Therefore, in No Query, Only a single module of FCs is utilized to convert a feature map of size of 256x7x7 to that of size of 1024. The *table 3* shows the effect of distinction of query vector and input vector.

Experiments result shows that our SUF model performs better than MaskTrackRCNN, our target model to defeat in the VIS task with AP 3.6 which is about 12% increments. We think the way we train, similar to inference and the way the SUF capture instance feature maps using LSTM are the factor of performance enhancement.

6. References

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

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshickm, Mask R-CNN. In ICCV, 2017.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In ICCV, 2015.

[4] Sepp Horcheiter, Jurgen Schmidhuber, Long-Short Term Memory. In Neural Computation, 1997.