Real-Time Partial Style Transfer

Miao Qi, Yu-Lin Shen

mq538@nyu.edu, yls247@nyu.edu

Abstract. The project aims to real-time processing the result on style transfer and object segmentation on the human. We leverage the model techniques, Fast Neural Transfer [8] and Yolact [1]. We constructed the pipeline function to concatenate two models and built the presentation using the webcam to real-time style transfer human while using video call.

1 Introduction

Among the current project online, style transfer is one branch that people will choose as the project topic. Most of the projects focus more on the style outcome and present a nice result. This is great, but we are thinking that whether there are other applications that can be utilized.

During the pandemic recently, the request for online meetings has increased. In the online meeting, we can add a mask to our background or modify it with a customized image. That brings an idea to use to our project. Considering the scenario that the user wants to make themselves to become another style in the famous art painting or live in the environment in the famous art painting but without changing the detail of their own environment, just change the style of it. This will involve the segmentation technique and real-time webcam processing. We combined these techniques and present them in this project.

For the following outline, we will briefly introduce the relevant technique to the style transfer and object segmentation in the Related Work. Present the demo result and explain the technique that we're using in this project in Implementation & Result. Finally, sharing some conclusions and future works.

2 Related Work

2.1 A Neural Algorithm of Artistic Style

Style transfer is originated from Gatys [4]. In the paper, they find that while using the CNN, down sampling will reduce the size of the image pixel, which leads to a decrease in the network unit. This is the way to reconstruct the image. With

the reconstruction method, to reach the goal of the style transfer, they provide the reconstruction of the content image and extract the style of the style image. They use the VGG network [11] as the experiment.

In the VGG network, there are five substructure, conv1 (a), conv2 (b), conv3 (c), conv4 (d) and conv5 (e). In the style image reconstruction, they find that the result in the lower layer of the a, b, and c, can be capture clearly; in the higher layer, d and e can preserve the content nicely. This brings the idea of designing content loss function and style loss function .

The loss function is listed as below:

$$L_{content} = \frac{1}{2}\sum_{i,j}(F_{i,j}^{l} - P_{i,j}^{l})^2 \tag{1}$$

$$L_{style} = \sum_{l=0j}^{L} w_l E_l \tag{2}$$

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(G_{ij}^{l} - A_{ij}^{l})^2 \tag{3}$$

$$G_{i,j}^{i} = \sum_{k} F_{ik}^{l} F_{kj}^{l} \tag{4}$$

The content loss function is listed as (1), P and F are the featured representative of the original image and the generated image. It is used to keep the detail of the original image. For the style loss function, it uses the concept of the Gram Matrix, (4). Gram Matrix is the dot product of the different feature representatives in VGG. In function (3), A is the original feature representative. The style loss function (2) is the weighted sum of the (3). This explains the style loss consists of the difference between the Gram Matrix, G, and A, which is a way to preserve the style. With the two-loss function, the team reconstruction the image with the new style.

2.2 Mask R-CNN

Mask R-CNN [7] is the technique using on object detection and segmentation. The developing history for the technique is from R-CNN [6], Fast R-CNN [5], Faster R-CNN [10], and finally to the Mask R-CNN. For the R-CNN family, the main technique is to utilize the Region Proposal technique with the CNN network to present the result. Region Proposal is the traditional computer vision technique that blocks the potential objects. We won't talk more about the Region Proposal and only focus on the Mask R-CNN here.

For the model, Mask R-CNN, with the input image, it will return the object information, including the class label, bounding boxes, and the edge of the object, which is the mask. The model uses a CNN network, the backbone can be VGG or Resnet, to extract a feature map as a start. The feature map uses the RPN (Region proposal network) to get the potential ROI (Region of Interest). Next, using ROIAlign map the input image pixel with ROI pixel. Finally, using these ROIs to make the classification, bounding box regression, and mask generated. For the mask generated, it processes the FCN in each pixel. FCN is the way to generate the segmentation. It will downsample the image size and adding the channel, which represents the features. After all, upsampling back to get the segmentation image.

2.3 Yolo

Yolo [9] is the technique that uses to find specific object information, including the class label and the bounding box. Unlike the region-based technique in the R-CNN family, which finds the potential object in advance, Yolo separates the image from the grid. The only request is to find the bounding boxes that contain the center of the object on the image. It output multiple images with the confidence score as well. In the confidence score, it uses the IOU (intersection over union) to capture the confidence. Since no Region Proposal technique is involved, it is way faster than the normal R-CNN family, but the accuracy might decrease since it only requires the output bounding box of the model contain the center of the object.

They are a bunch of transformations and versions in the Yolo family and we might not introduce all of them in this project. We just want to give the concept difference of choosing the Yolo or R-CNN family, which is the trade-off between the speed and the accuracy.

3 Implementation & Result

In this project, both the style transfer model and segmentation model are pretrained, the dataset is from the COCO dataset. Our concentration is to combine the two models and create the partial style transfer effect. We have researched the details of the model technique and utilizing TorchScript for the implementation.

3.1 Fast Neural Transfer

Although [4] provides a nice idea on style transfer originally, the processing time to run through the model is costly. In [8], it hugely accelerates this processing from 2 seconds to 10 mill seconds. The critical modification is adding the Image Transform Net before using the VGG backbone on the style transfer. It basically is thinking of as an auto-encoder, which using down sampling and up sampling techniques. In the paper, there are two benefits of using this. First, working the convolution in the downsampling can sharply decrease the computation compared to the original paper in [4], which processing the convolution on the image that shaped in the original size. Second, to produce a high-quality style transferred image, it will change a lot on the original image simultaneously. This means that it will increase the effective receptive field. With the downsampling, it can use the same size of the layer to larger the effective receptive field.

3.2 Yolact

Although Mask R-CNN [7] presents an accurate result on capturing the object on the image, especially on the multiple objects, still, it needs lots of time to process. The main reason is the two steps segmentation, which detects the RPN (Region Proposal Network) to generate the ROI (Region of Interests) and further using the ROI on the classification and segmentation. In the Yolact [1], the author believes that using fully connected to predict the classification and using the convolution to gather the edge is the better choice. Note that for the edge block of the image, the paper also provides that capturing the mask and mask coefficient separately may accelerate the process as well. Since the model didn't use the region proposal technique, it sacrificed a few accuracies, but it extracts the way faster speed on getting the output.

For the structure, it uses a pre-trained CNN backbone (RetinaNet) and the FPN (Feature Pyramid Networks) to increase the capturing ability on the small object. Then it processes the output into two branches: the Prediction Head and Prototype masks. In the Prediction Head, it outputs label class, bounding box, and mask coefficients. For the mask coefficients, it uses the tanh as the activation function to giving more combinations on the object prediction (-1 and 1). For the Prototype masks, it input the P3 from the output of FPN. Only combining the two branches can generate the instance

3.3 TorchScript

TorchScript is the technique to use the model in an object-oriented way. Normally, in PyTorch, using a model requires first import the model architecture

and then load the model weights. If there is a customize setting either on the library or the code, it needs to be configured as well while using the model. With the jit and trace in the TorchScript, everything is wrapped in the script. The saved model can be called without any extra loading model weight or library. It can be saved in the script once using TorchScript. This saves a huge time while calling the model in our real-time pipeline. One thing to note that is the input and output format using the TorchScript. During the implementation, the TorchScipt has not supported the bool input and the dictionary output for know. Users may use other techniques to deal with it, such as argparse. Here, since our input and output are both tensors, there won't be a problem.

3.4 Demo

We use the Jennifer Aniston online photo as the sample. The result of different style is listed as the Fig.


Fig. 1. princess


Fig. 2. udnie


Fig. 3. mosaic


Fig. 4. candy


Fig. 5. multiple

We also construct multiple styles on different objects as well. In the Fig. 5, the background is transferred in one style and the human object is transferred to another style.

For the real-time video, the code is saved in the Github. However, due to the resource, we combine two models sequentially. This will affect the performance of real-time. Note that for image size from 510 to 512, Yolact baseline is tested as 42 fps and neural fast style transfer is 20 fps. Although it may not reach real-time processing (30 fps), still, it should still process the video with 14 fps. On the colab, for both the style model and Yolact model, it typically takes 0.5 second to run, which not as fast as we expected. We assume that this might because of the limited resource on the colab and the internet. We still provide the code to run the demo.

4 Conclusion & Future work

In this project, we produce the real-time localize style transfer result that using the webcam. Although the results capture and transfer nicely, still, if the user wants to modify any object with a different style in real time, this project cannot satisfy the need. Since the original thought from the [8], during the training, one style model can be generated at once. With the current method, multi-style require multi-models. To reach the goal of adding a different style to a different object, it will use a huge resource on GPU while using the pipeline. There is one way to resolve this issue is [2]. It separates the step to generate the style and the content. While outputting, different style maps to different layers then produce the image. Any other issue will be adding a new style require the new training task. [3] is used to create an image with any style input instantly. Both methods can be the new target for this project and are helpful to generate the GUI interface playing on the different objects in real-time if needed.

References

1. Bolya, D., Zhou, C., Xiao, F., Lee, Y.J.: YOLACT: real-time instance segmentation. CoRR **abs/1904.02689** (2019), http://arxiv.org/abs/1904.02689
2. Chen, D., Yuan, L., Liao, J., Yu, N., Hua, G.: Stylebank: An explicit representation for neural image style transfer. CoRR **abs/1703.09210** (2017), http://arxiv.org/abs/1703.09210
3. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. CoRR **abs/1610.07629** (2016), http://arxiv.org/abs/1610.07629
4. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. CoRR **abs/1508.06576** (2015), http://arxiv.org/abs/1508.06576
5. Girshick, R.B.: Fast R-CNN. CoRR **abs/1504.08083** (2015), http://arxiv.org/abs/1504.08083
6. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR **abs/1311.2524** (2013), http://arxiv.org/abs/1311.2524
7. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. CoRR **abs/1703.06870** (2017), http://arxiv.org/abs/1703.06870
8. Johnson, J., Alahi, A., Li, F.: Perceptual losses for real-time style transfer and super-resolution. CoRR **abs/1603.08155** (2016), http://arxiv.org/abs/1603.08155
9. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. CoRR **abs/1506.02640** (2015), http://arxiv.org/abs/1506.02640
10. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR **abs/1506.01497** (2015), http://arxiv.org/abs/1506.01497

11. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014), http://arxiv.org/abs/1409.1556