Project Report - ECE 285

Unay Shah
Electrical and Computer Engineering
A59015777

Prithwiraj Paul
Electrical and Computer Engineering
A59020278

Abstract

The objective of the proposed project is to overcome a limitation found in current style transfer methods utilized in the field of Computer Vision. Style transfer is commonly used to generate images that depict how a given scene would look if it were created by a particular artist, altered with specific filters, or captured in specific conditions. However, existing techniques typically apply a single style to the entire image, disregarding the potential to assign distinct styles to different regions within the same image. The primary goal of this project is to develop a pipeline that can assign different styles to various elements present in a single image. The implementation can be found in the GitHub Repository.

1 Introduction

Class-based style transfer has gained significant popularity for its ability to create visually appealing art, facilitate photo editing, and enhance fashion designs. By modifying specific elements within an image, it offers artists the flexibility to express their creativity. However, existing style transfer methods typically apply a uniform style to the entire image, limiting the artistic exploration of assigning different styles to different regions. This project aims to address this limitation by developing an efficient pipeline that combines semantic segmentation with fast style transfer to assign distinct styles to various elements within an image.

The motivation behind this project stems from the significance of style transfer and semantic segmentation in the field of computer vision. Style transfer techniques allow us to explore how an image would appear if painted in the style of renowned artists or subjected to specific artistic filters. It serves as a powerful tool for creating visually appealing artwork, facilitating photo editing, and enhancing fashion designs.

Semantic segmentation, on the other hand, plays a crucial role in object recognition and scene understanding by accurately segmenting an image into distinct objects or regions. This segmentation provides a foundation for assigning specific styles to individual objects within an image, enabling precise control over their visual appearance.

By combining the capabilities of fast style transfer and semantic segmentation, we aim to achieve a seamless integration of these two processes. The goal is to develop a pipeline that not only assigns different styles to different objects or regions but also performs this operation at a high speed. Real-time performance is a crucial aspect to enable dynamic and interactive applications of style transfer, targeting a speed of 25-30 frames per second (fps).

The integration of fast semantic segmentation with style transfer allows us to overcome the limitation of applying a uniform style to the entire image. By segmenting the image into objects or regions and assigning specific styles to each segment, we can achieve a more nuanced and visually captivating result. This capability opens up exciting possibilities for artistic expression, visual storytelling, and collaborative creation, as each object can be stylized according to its unique characteristics.

The proposed solution comprises two essential components. Firstly, an extensive literature survey was conducted to explore the available Fast Semantic Segmentation Models. Among them, we

carefully selected the Fast-SCNN semantic segmentation model to achieve efficient and rapid semantic segmentation output. The Fast-SCNN model was pretrained on the CityScape dataset, which provides a diverse and annotated collection of urban street scenes.

Secondly, our focus shifted to the task of Fast Neural Style Transfer. Our main objective in this project was to enhance the fast style network to accurately transfer styles to content images. To train the Fast Neural Style Transfer model, we utilized two types of inputs: style images and content images. The style image set consisted of nine different style images created by various artists, including multiple styles from the same artist. As for content images, we utilized the widely recognized PascalVOC2012 dataset, which encompasses a broad range of object categories.

Upon training the model, we ensured its real-time performance by utilizing a pre-trained version. This allowed for smooth and interactive style application during the transfer process. To combine both the semantic segmentation model and the style transfer model, we established a pipeline. This pipeline serves as a foundation for efficiently transferring different styles to the segmented elements within an image. Additionally, we customized the architecture of the Fast Neural Style Transfer model to further improve its performance.

In order to evaluate the efficacy of our approach, we conducted a qualitative study. We applied the style transfer process to images from the CityScape and PascalVOC datasets and analyzed the results produced by our model. This qualitative analysis enabled us to assess the quality of style transfer and verify the alignment between the assigned styles and the semantic segmentation of the images.

By integrating state-of-the-art Fast Semantic Segmentation Models and our enhanced Fast Neural Style Transfer technique, our proposed pipeline offers a powerful solution for generating visually captivating compositions with specific artistic styles. The combination of these components provides a seamless and efficient process, unlocking new possibilities for creative image manipulation and composition.

The expected results include obtaining a foreground mask using FAST-SCNN on the background image, generating multiple stylized images of the content image using custom Neural Style Transfer Models, and merging object-specific styles from the semantic segmentation masked image to produce the final stylized image. We hope to build models that both run at about 20-25 fps, so that the combined pipeline can process images in real-time.

The report is structured into five sections. Section I provides an extensive introduction to the problem at hand, outlining its significance and scope. Section II presents a concise overview of the existing research and approaches related to this problem. Section III outlines the technical approach adopted to tackle this problem, describing the methods and algorithms employed. Section IV details the experiments conducted and the obtained results in addressing the problem. Finally, Section V discusses the conclusion, and provides an overall summary of the project.

2 Related Work

In this section, we describe and compare two fast image segmentation models Fast-SCNN and DABNet with a particular focus on real-time execution with low energy and memory requirements. Following that, we describe the conventional Style Transfer models and how the training algorithm is modified to generate fast style transfer in real-time.

2.1 Style Transfer

In the realm of style transfer, the seminal work by Gatys et al. [1] introduced the use of deep convolutional neural networks (CNNs) to achieve whole-image styling. By simultaneously minimizing both the content and style loss, they were able to produce visually stunning results. However, their approach suffered from relatively slow inference speeds, making it impractical for real-time video applications. To address this limitation, Johnson et al. [2] extended the work of Gatys et al. by introducing a feed-forward network. This advancement significantly improved the inference speed, enabling real-time video style transfer. Their approach allowed for the efficient application of style to videos, facilitating a more practical and interactive user experience.

2.2 Fast Semantic Segmentation

In the domain of real-time semantic segmentation, a key challenge is balancing accuracy with inference speed. Many existing frameworks designed for single-image semantic segmentation struggle to perform well in real-time due to their slow inference times.

Efforts have been made to optimize the speed of semantic segmentation networks. For example, Li et al. [3] proposed a solution to this problem by introducing the Depthwise Asymmetric Bottleneck module (DABNet). This module efficiently extracts both local and contextual features, thereby enhancing the speed of semantic segmentation without compromising accuracy. Their approach successfully addressed the speed constraints associated with real-time semantic segmentation, providing a valuable contribution to the field.

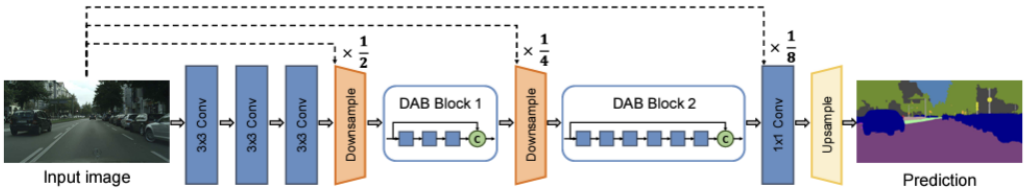


Figure 1: DABNET Model Architecture [3]

Another notable fast semantic segmentation model is FAST-SCNN (Fast Segmentation Convolutional Neural Network) [4]. It heavily relies on depthwise separable convolutions and residual bottleneck blocks to achieve efficient segmentation. Additionally, FAST-SCNN introduces a two-branch model that incorporates a "learning to downsample" module, enabling shared feature extraction at multiple resolution levels. This approach takes advantage of similar features extracted in the initial layers of the multiple branches. With its combined features, FAST-SCNN achieves fast and accurate semantic segmentation results.

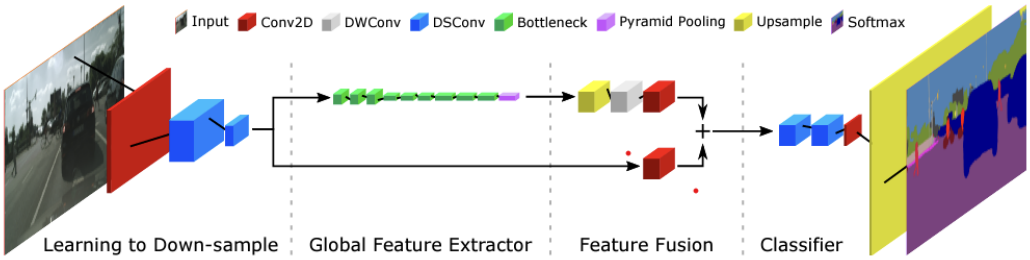


Figure 2: FAST SCNN Model Architecture [4]

By incorporating techniques from these fast semantic segmentation models, we aim to achieve real-time style transfer while maintaining high accuracy in the segmentation results. Comparing the performance metrics provided for DABNet and Fast-SCNN on the Cityscapes dataset, we can observe the following:

- **DABNet**: Without any pretrained model or postprocessing, DABNet achieves a Mean Intersection over Union (Mean IoU) of 70.1% on the Cityscapes test dataset. It accomplishes this with only 0.76 million parameters and operates at a speed of 104 frames per second on a single GTX 1080Ti card.
- **Fast-SCNN**: On the other hand, Fast-SCNN combines spatial details at high resolution with deep features extracted at lower resolution. This approach yields an accuracy of 68.0% Mean Intersection over Union on the Cityscapes dataset. It achieves this performance while operating at a speed of 123.5 frames per second.

Based on these metrics, DABNet demonstrates a slightly higher accuracy with a Mean IoU of 70.1% compared to Fast-SCNN's 68.0%. However, Fast-SCNN offers a higher speed of 123.5 frames per second, whereas DABNet operates at 104 frames per second. Additionally, DABNet has a smaller model size with 0.76 million parameters compared to Fast-SCNN, but it's worth noting that pretrained models are not used in the reported performance.

2.3 Fast Style Transfer

In their work on localized real-time style transfer, Castillo et al. [5] introduced a method that focuses on applying neural style transfer (NST) to specific regions within still images. They achieved this by masking the objects of interest in the target image and modifying the loss function to consider only the pixels within those regions. By doing so, they were able to create images with partial styling.

To determine the foreground and background pixels, Castillo et al. employed a pixel classification model. They then applied the style to the pixels corresponding to the selected objects, allowing users to specify the regions for styling. In order to address the issue of boundary artifacts and improve the visual quality, they utilized Markov random fields (MRFs) to smooth out the outlier pixels.

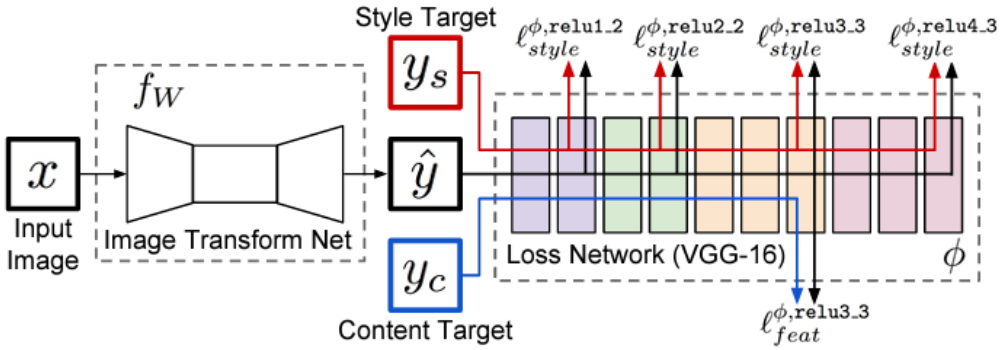


Figure 3: Neural Style Transfer Model

However, their approach did not achieve real-time performance due to their reliance on the slower method introduced by Gatys et al. [1] for style transfer. The iterative nature of Gatys et al.'s method limited the speed of the style transfer process. Another proposed method by Kurzman et. al [6], introduced CBS (Class Based Style) that enables real-time style transfer to specific object classes. To perform style transfer on an image I for a particular object class C, CBS follows these steps:

- Using a fast segmentation network, CBS extracts a binary mask Rc, which identifies pixels belonging to object class c. Rc has the same dimensions as I, with entries set to 1 for pixels belonging to class c and 0 for others. It's important to note that the segmentation network is pre-trained to segment object class c.
- Simultaneously, CBS transforms the input image I into a styled image TS using a fast styling method (such as the approach introduced by Johnson et al. [2]).
- Utilizing the binary mask Rc, CBS extracts the background Ib from the unstyled image I and the foreground Tf from the styled image TS.
- Finally, CBS combines Tf and Ib to obtain a target image U, which consists of an unstyled background and a styled object class c (as depicted in Figure 3).

While this pipeline shares similarities with Castillo et al.'s approach, their method is slow due to the reliance on Gatys et al.'s method [1], Mask R-CNN, and MRF blending. In contrast, the CBS framework can achieve real-time performance at a speed of 16 frames per second (fps).

3 Method

In this section, we present our methodology for addressing the fast style transfer problem and outline the technical approach used to construct the pipeline and design the architecture of our custom style transfer model.

3.1 Background

3.1.1 Fast Semantic Segmentation

Drawing from the performance metrics of DABNet and FAST-SCNN Network for fast semantic segmentation mentioned in Section 2, we have selected the FAST-SCNN model for our fast-style transfer project. The choice of DABNet and Fast-SCNN model depends on factors such as the desired balance between accuracy and speed, available computational resources, and the importance of model size. If achieving a higher accuracy is crucial and computational resources permit, DABNet might be a suitable choice. On the other hand, if real-time performance is a priority while still maintaining good accuracy, Fast-SCNN's faster inference speed could make it more appealing. For our application, real-time style transfer performance is the priority since it has to take fast inference, hence we selected FAST-SCNN model for further analysis.

Expanding upon existing two-branch methods for fast segmentation, FAST-SCNN network introduces a novel "learning to downsample" module. This module enables simultaneous computation of low-level features for multiple resolution branches and has a single skip connection. By combining spatial details at high resolution with deep features extracted at a lower resolution, this network achieves an impressive accuracy of 68.0% mean intersection over union on the Cityscapes dataset, all while operating at a blazing speed of 123.5 frames per second.

3.1.2 Fast Neural Style Transfer

In our project, we adopt the CBS model architecture as a baseline. However, we aim to improve upon the inference time. Instead of modifying the content image iteratively, as done in the conventional CBS method, we propose to accelerate the process by learning and storing the weights. This allows us to perform style transfer iteratively using a single style image and multiple content images.

By employing fast semantic segmentation and fast style transfer techniques, we can significantly speed up the style transfer process for multiple images. The model learns from diverse types of images and minimizes losses for each image, enabling the transfer of styles to a wide variety of images without explicitly requiring the style image as an input. By integrating fast semantic segmentation and fast style transfer, we can achieve efficient and real-time style transfer for multiple images.

Finally, we propose to make a pipeline where we can achieve localized and diverse style transfer in real-time. By integrating the FAST-SCNN semantic segmentation model and a pre-trained Neural Style Transfer model, we can effectively combine the benefits of fast semantic segmentation and fast style transfer.

The pipeline (Figure 4) is described as follows:

- **Input**: Start with a cityscape image.
- **Fast Semantic Segmentation**: Utilize the FAST-SCNN model to obtain a masked segmented image. This step identifies the different objects or regions within the image.
- **Style Application**: For each segmented object in the image, apply a different style using the corresponding full stylized image from step 3. This allows us to transfer a variety of styles to different objects within the cityscape image.
- **Output**: The final output will be a fully stylized image where each segmented object has a different style applied.

For the project implementation, we mostly used Google Colaboratory (Free Version) and UCSD's Jupyter Datahub platform, both of which have restricted GPU resources. As a result, the training for the FAST-SCNN model takes a lot of time due to lack of hardware resources for executing extensive computation and this would surpass the CUDA RAM and crash our notebook. As a result, we had to use the pre-trained weights for FAST-FCNN model for semantic segmentation. However,our focus

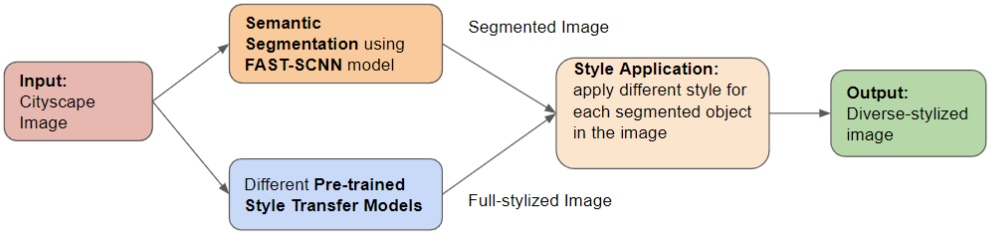


Figure 4: Integrated Pipeline Block Diagram for Fast Style Transfer

was moslt on Fast Neural Style Transfer model and customizing the model. We trained the Fast Neural Style Transfer model on Collab and our custom Fast Neural Style Transfer model for multiple content images and multiple style images. We used popular artwork from multiple artists for style transfer. We tried to take multiple paintings by a single artist, and tried to learn their style from all the artworks simultaneously.

3.2 Details of the Model used

3.2.1 Mathematical Formulation of Fast Style Transfer

Our model's objective is to generate real-time stylized images that combine the content of an input image (I) with the style of another image (S). We utilize the fast styling method (FSM) introduced by Johnson et al. [2], which employs a fully convolutional neural network (FCN) to learn how to generate stylized images. During training, the FSM optimizes a loss function (equation 1) that includes a content loss and a style loss.

The content loss ensures that the generated stylized image retains the spatial structure of the input image, while not excessively focusing on preserving its color, texture, or exact shape. This loss is computed by comparing the feature representations of the input image and the generated image using the L2 norm.

$$L(I) = \frac{1}{C_2 \cdot H_2 \cdot W_2} \cdot \|F_2(M(I)) - F_2(I)\|_2^2 + \sum_{l=1}^{L} \frac{1}{C^l} \cdot \|G(M(I), l) - G(S, l)\|_F^2 \quad (1)$$

$$ContentLoss = \cdot \|F_2(M(I)) - F_2(I)\|_2^2$$

$$StyleLoss = \sum_{l=1}^{L} \frac{1}{C_l} \cdot \|G(M(I), l) - G(S, l)\|_F^2$$

$$G_{i,j}(X, l) = \frac{1}{H_l W_l} \sum_{h=1}^{H_l} \sum_{w=1}^{W_l} F_{l(h,w,i)}(X) \cdot F_{l(h,w,j)}(X) \quad (2)$$

The style loss aims to preserve the style characteristics, such as color and texture, from a style image. It is computed by evaluating the uncentered covariance between the extracted features of an image, represented as a Gram matrix (equation 2). The Gram matrix captures the relationships between feature pairs and measures the tendency of feature channels to activate together.

By optimizing the combined content and style losses, our model learns to generate stylized images that maintain the spatial structure of the input image while incorporating the desired style from the style image. A key advantage of our approach is that it achieves real-time generation of stylized images by performing a single forward pass of the FCN on the input image, eliminating the need for iterative optimization.

In the final step of style transfer, our objective is to generate an image (I) that contains different stylized object classes in the image. To achieve this, we follow a two-step process.

- First, we obtain the foreground mask for the background image (content image) by using the FAST-SCNN model. This mask represents the regions corresponding to the object classes in the content image. Parallelly, we obtain the stylized image of the content image using the custom-design Neural Style Transfer Model

- Next, we obtain the final stylized image by merging all the object-specific style (object index and features obtained from the semantic segmentation masked image).

3.2.2 Network Architecture Implemented

Fast Style Transfer

The model consists of two major units: an encoder-decoder style network and a pretrained VGG16 network. The encoder-decoder network consists of a set of convolutional layers, residual connections and upsampling convolutional layers. The encoder learns high level features from input images. The residual blocks try to learn finer details from both the style image and content image. The upsampling layers try to rebuild the style image from the images produced by previous layers.

The authors use Instance Normalization (IN) layers instead of Batch Normalization (BN) layers. The advantage of using IN is that features are sharper and localized, along with holding good contrast. BN results in blurry images and low contrast.

After the encoder-decoder network, a pre-trained VGG16 network is used to extract features from the output image. These features are useful to identify objects in the image, so that they can be preserved, maintain their structure and hold good contrast with their surroundings. Using pre-trained weights helps improve the model's performance, as it already gets good features into the loss functions from the first epoch. A combination of content loss and style loss (using gram matrix) are used to improve the style transfer, while preserving the original image.

For training, we randomly select 100 images from PascalVOC dataset and one or more style image. These are trained with a batch size of 25 for 300 epochs. We used a learning rate of 0.001, content weight of 10^5 and style weight of 10^{10} (Table 1). These weights are scale factors to consider when calculating the losses. Testing for this network is more qualitative, with only the speed being a measurable value of importance.

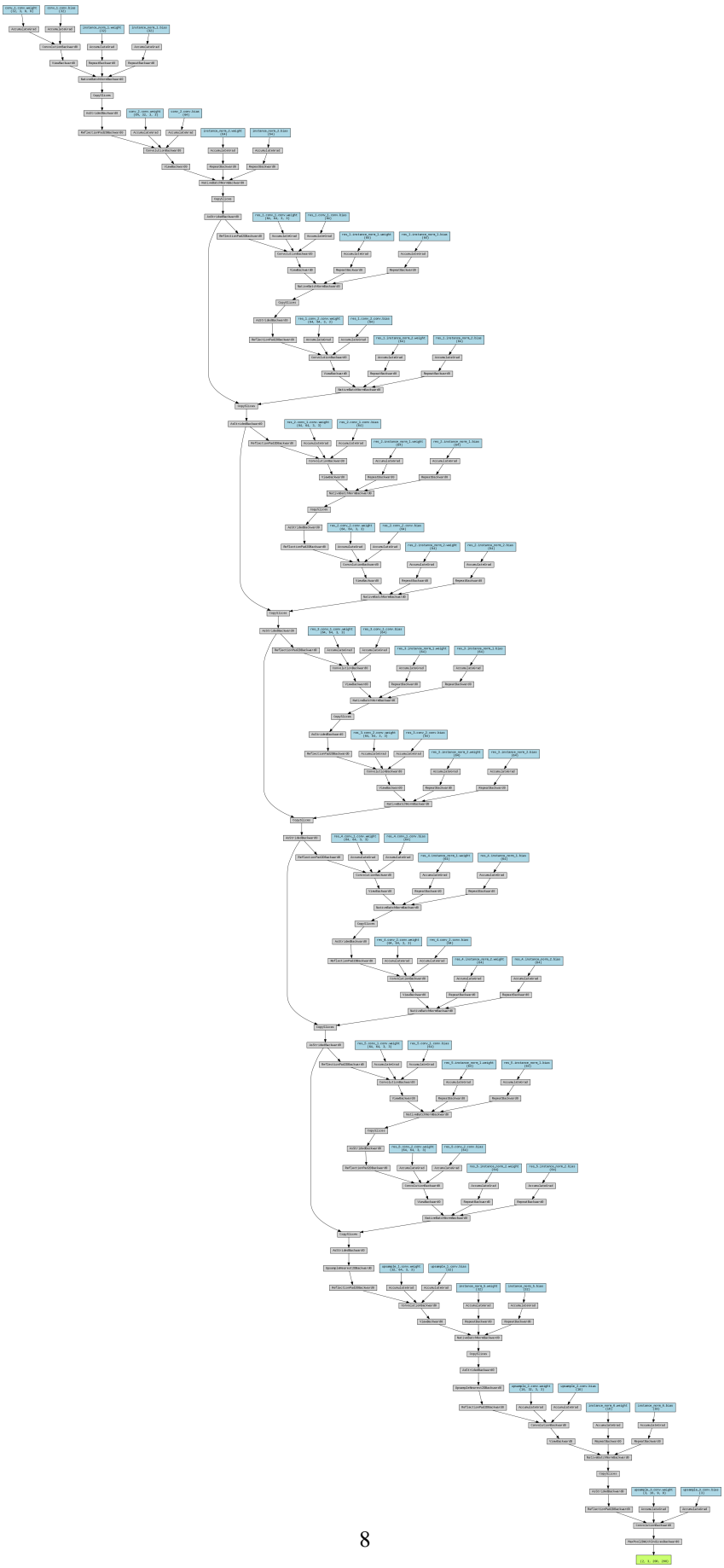


Figure 5: Neural Style Transfer Model

Table 1: Training Parameters for Fast Style Transfer Model

Parameter	Value (Single Style)	Value (Multiple Style)
Number of Images	100	100
Style Image	1	> 1
Batch Size	25	25
Number of Epochs	300	500
Learning Rate	0.001	0.001
Content Weight	10^5	10^5
Style Weight	10^{10}	10^{10}
Optimizer	Adam	Adam

We tried various modifications for the network and finally arrived at a model by removing some convolutional layers and using smaller sized residual layers (Figure 5). We removed 1 layer from the encoder network, reducing the output filter size of this part of the network to 64 filters instead of 128 as used by the original paper. This also led to using residual network blocks of size 64, and one step for upsampling. The decoder network now upsamples the image from 64 filters instead of using 128 layers. The reason for modifying this was to decrease the model size, improve speed and understand whether scaling the model down would hinder the results. But this led to having a model which was $1/4^{th}$ of the original model, retaining more features from the original image while also transferring more elements from the style image.

Fast SCNN

FAST-SCNN (Fast Segmentation Convolutional Neural Network) is a semantic segmentation model that emphasizes efficient segmentation by utilizing depthwise separable convolutions and a two-branch architecture. It consists of four main components: the Learning to Downsample module, the Global Feature Extractor, the Feature Fusion module, and the Classifier.

- The **Learning to Downsample module** employs three layers to learn downsampling operations and facilitate low-level feature sharing. It includes a standard convolutional layer (Conv2D) followed by two depthwise separable convolutional layers (DSConv). By using a stride of 2 for downsampling, followed by batch normalization and ReLU activation, this module ensures efficient feature extraction.

- The **Global Feature Extractor** captures global context information for segmentation by directly operating on the output of the Learning to Downsample module, which is at 1/8-resolution of the input. It utilizes efficient bottleneck residual blocks, inspired by MobileNet-V2, and incorporates residual connections for blocks with matching input and output sizes. Depthwise separable convolutions are employed to reduce parameters and floating-point operations. The addition of a pyramid pooling module (PPM) aggregates context information from various regions.

- The **Feature Fusion module** employs a simple addition-based fusion approach, similar to ICNet and ContextNet. It combines the fused features from the Global Feature Extractor and the shallow branch, ensuring efficiency. Non-linearity is applied after the features are added.

- The **Classifier module** performs pixel-wise classification to generate the final segmentation map. It includes two depthwise separable convolutions (DSConv) and one pointwise convolution (Conv2D). Additional layers are added after the Feature Fusion module to enhance accuracy. During training, softmax activation is used, while during inference, the computationally expensive softmax computations can be substituted with argmax, providing a faster alternative (denoted as Fast-SCNNcls). Alternatively, softmax can be used for a standard probabilistic model (Fast-SCNNprob).

Overall, FAST-SCNN effectively combines depthwise separable convolutions, shared feature extraction, and a two-branch architecture to achieve fast and accurate semantic segmentation. It leverages low-level feature sharing, captures global context information, and employs an efficient fusion approach, resulting in efficient and reliable segmentation results.

We tried to modify the network architecture but due to lack of GPU resources we were not able to train the model so we re-implemented the same architecture from the paper. The training parameters used to train the FAST-SCNN model are mentioned in Table 2.

Table 2: Training Parameters for FAST-SCNN Model

Parameter	**Value**
Number of Epochs	100
Batch Size	50
Number of Images	200
Dataset	Cityscape
Learning Rate	0.01
Learning Rate Scheduler	Polynomial Decay
Decay Power	0.9
Starting Learning Rate	0.01
Optimizer	Adam

4 Experiments

4.1 Datasets

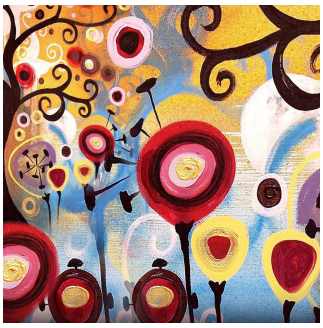
To provide quantitative and qualitative insights into the performance of our Fast Style Transfer model, we evaluate the performance of our model on the following two datasets:

- **Cityscape**: The Cityscapes Dataset [7] focuses on semantic understanding of urban street scenes. It provides semantic, instance-wise, and dense pixel annotations for 30 classes grouped into 8 categories (flat surfaces, humans, vehicles, constructions, objects, nature, sky, and void). The Cityscape dataset serves as our primary dataset, and we use it for performing both qualitative analysis, as well as the ablation study of our model. The Cityscapes dataset consists of 2,975 training images, 500 validation images, and 1,525 labeled and 1,525 unlabeled testing images.
- **WikiArt**: WikiArt dataset [8] contains paintings from 195 different artists. The dataset has 42129 images for training and 10628 images for testing. The WikiArt dataset comprises a collection of artwork images, primarily stored in JPEG or PNG format. The images are organized into directories, often based on criteria such as artist, style, or genre. In addition to the image files, the dataset includes accompanying metadata, such as the artwork's title, artist name, style, and genre. This metadata is commonly provided in structured formats like CSV, JSON, or XML, allowing for further exploration and analysis of the artworks within the dataset.
- **VOCDataset2012**: The VOC dataset 2012 [9] is a benchmark dataset widely used in computer vision for tasks like object detection, semantic segmentation, and image classification. It consists of around 11,540 images with annotated object bounding boxes and pixel-level semantic segmentation masks. With 20 object classes and train/val/test splits, it provides a diverse range of images for training style transfer models, allowing researchers to explore and develop innovative approaches in the field. The VOC dataset 2012 uses XML files to store annotations for object bounding boxes and pixel-level semantic segmentation masks. The images are typically in JPEG or PNG format.

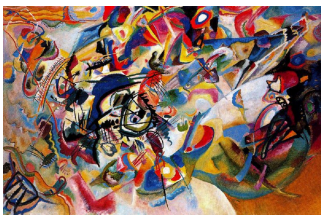
4.2 Results

4.3 Fast Style Transfer

Dataset: We use 100 images from PascalVOC dataset, stored in a folder without any associated labels. We use the following popular artworks:



(a) Candy (Cropped image from June Tree by Natasha Wescoat) [10]



(b) Composition VII [11]

(c) Girl Before a Mirror [12]

(d) Mosaic [2]

(e) Rain Princess [13]

(f) Starry Night [14]

(g) The Great Wave off Kanagawa [15]

(h) The Persistence of Memory [16]

(i) Udnie [17]

Figure 6: Style Images used for Single Style Transfer

(a) Impression V [18] (b) Several Circles [19] (c) Upward [20] (d) Yellow Blue Red [21]

Figure 7: Style Images used for Style Transfer of Artist Wassily Kandinsky

We also carry out various ablation studies using by using Batch Normalization, adding more layers to the encoder-decoder architecture with residual blocks of size 256 and removing layers from encoder-decoder architecture with residual blocks of size 64. Here are a summary of the results.

- **Batch Normalization** resulted in lesser style transfer and gave blurry results with low contrast
- **Increasing layers** did not have much effect on the output images
- **Decreasing layers** resulted in more style elements as well as content elements being visible in resulting image

(a) Original Image (b) Using Residual of Size 64

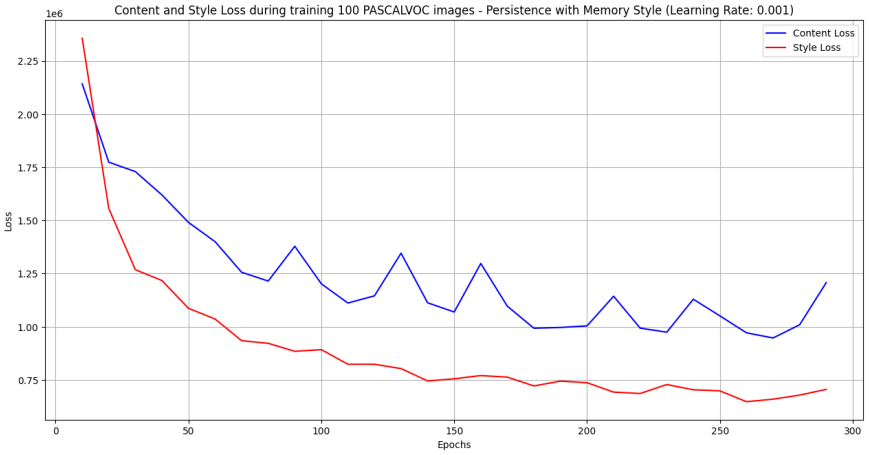
(c) Original Model (d) Using Batch Normalization (e) Using Residual of Size 256

Figure 8: Comparison of Models-1 (Udnie)

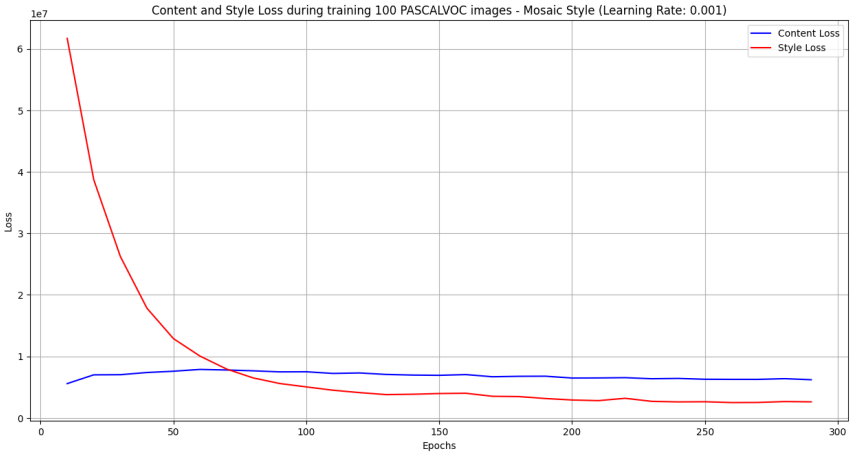
(a) Original Image (b) Using Residual of Size 64

(c) Original Model (d) Using Batch Normalization (e) Using Residual of Size 256

Figure 9: Comparison of Models-2 (The Great Wave off Kanagawa)



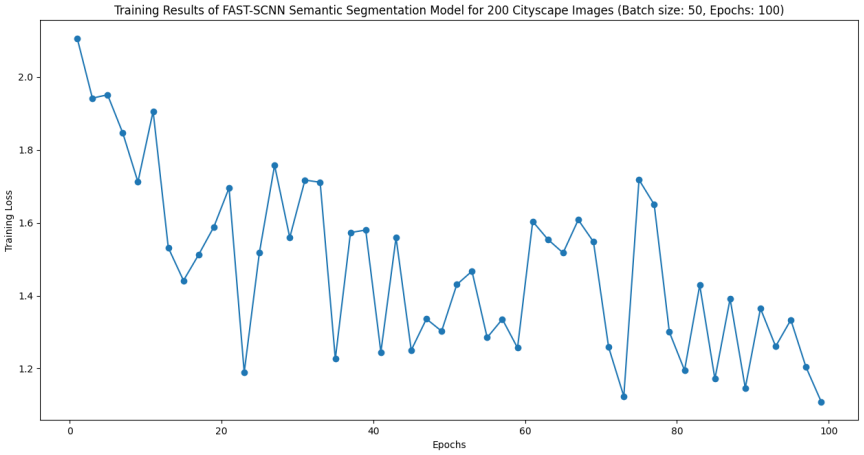
(a) Losses for Training Images with Artwork The Persistence of Memory



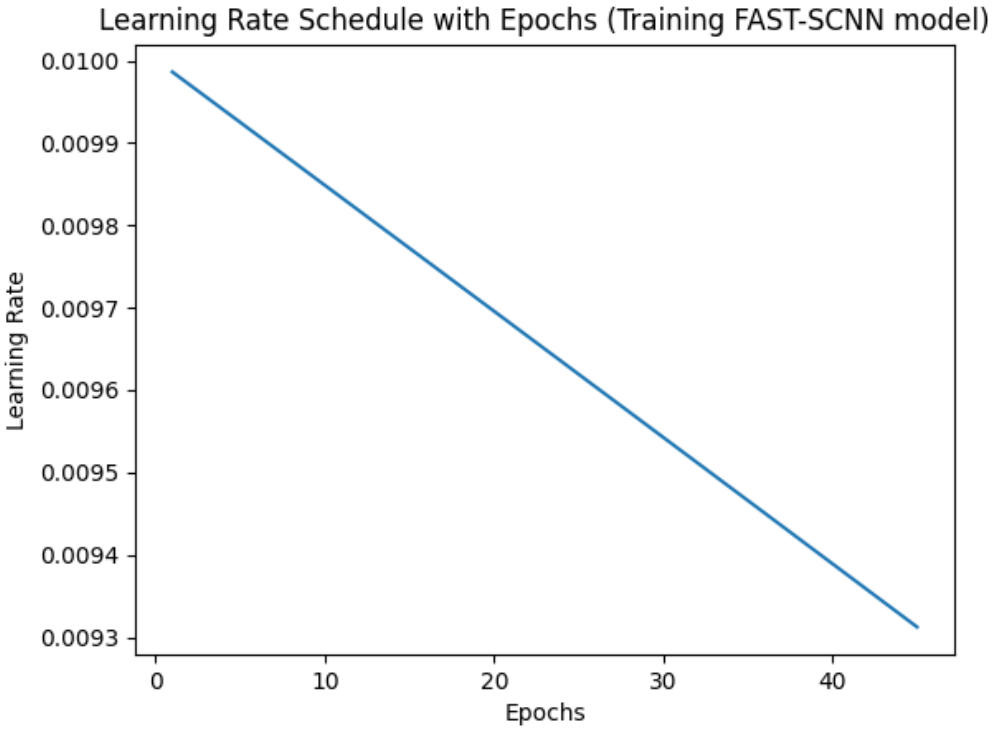
(b) Losses for Training Images with Artwork Mosaic

4.4 Fast SCNN

We trained Fast SCNN with 100 epochs on 200 images from Cityscape dataset.



(c) Training Loss



(d) Learning Rate

Figure 10: Training Metrics for Fast SCNN

The following observations and inferences were made based on the training results plots:

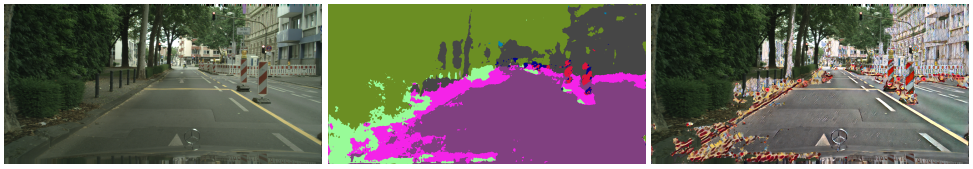
- **Adam Optimizer**: Adam (Adaptive Moment Estimation) is an optimization algorithm commonly used for training deep neural networks. It combines the benefits of both the AdaGrad and RMSProp algorithms. Adam adjusts the learning rate dynamically for each parameter based on the estimates of the first and second moments of the gradients.
- **Non-linear Training Loss**: The non-linear decrease in the training loss can be attributed to the Adam optimizer's adaptive learning rate. Adam calculates different learning rates for each parameter based on the first and second moments of the gradients. This adaptivity allows the optimizer to handle different parameter updates effectively, leading to non-linear changes in the loss function over epochs.
- **Linear Learning Rate Decay**: Although the learning rate is decreasing linearly with epochs, the adaptive nature of Adam optimizer compensates for the fixed linear decay. The optimizer adjusts the learning rates dynamically for each parameter based on the first and second moments of the gradients. This dynamic adjustment helps the model to explore the loss landscape effectively, even with a linearly decreasing learning rate.
- **Benefits of Adam**: Adam optimizer is known for its efficient and effective optimization of neural network models. It combines the advantages of adaptive learning rates, momentum, and bias correction, resulting in faster convergence and better generalization. The adaptive learning rate helps the optimizer navigate different regions of the loss landscape, while the momentum term allows for faster updates in the relevant directions.

4.5 Results from the combined custom-made pipeline for real-time diverse style transfer

The results of the style transfer for two selected Cityscape images are shown below:

(a) Original content image (b) Segmented Image (c) Diverse Stylized Image

Figure 11: Style Transfer Output I



(a) Original content image (b) Segmented Image (c) Diverse Stylized Image

Figure 12: Style Transfer Output II

From Figure 11 and Figure 12, it is observed that our pipeline combines segmentation and style transfer techniques, but the segmentation results are not satisfactory and the speed of segmentation is slow at only 1 frame per second (fps). However, the style transfer component produces impressive output. Despite the limitations in segmentation accuracy and speed, the combined pipeline still delivers visually appealing results. The overall speed of the pipeline is coming out to be approx. 1 fps. The fast-style transfer model works at 20-25 fps which is the bottleneck for our model. We need to investigate and address the factors causing the subpar segmentation and slow speed to improve the overall performance of the pipeline. By focusing on enhancing the segmentation component, we aim to achieve both accurate segmentation masks and high-quality stylized outputs in real-time.

5 Conclusion

In conclusion, our custom-designed pipeline demonstrates the capability to achieve real-time localized style transfer by segmenting and styling each class separately in real-time. This opens up possibilities for creating more intricate videos, captivating works of art, and compelling product advertisements with tailored styling. Currently, the pipeline operates at a speed of 1 frames per second (fps), which is limited by the output of the FAST-SCNN model. The fast-style transfer model works at 20-25 fps which is the bottleneck for our model. However, there is room for further optimization to enhance both the speed and accuracy of the pipeline.

Our modified architecture for style transfer is training models which produce images similar in quality to images produced by popularly used style transfer methods on a single image. The trade-off of using a pre-trained model, that has its weights tuned to a few images is not apparent in the test results. This is a combination of having a dataset with a wide variety of images and using a feature extraction model like VGG16 in the pipeline.

One area for improvement lies in optimizing the FAST-FCNN model, as it currently does not provide the desired high fps output, resulting in suboptimal results. Addressing this limitation should be considered in the future scope of our work. By enhancing the speed and efficiency of the pipeline, we can unlock its full potential for real-time style transfer in various applications.

Overall, our custom pipeline represents a significant step towards real-time localized style transfer and offers promising opportunities for creating visually appealing and engaging content. With further refinement and optimization, we aim to push the boundaries of this technology, enabling even more impressive and immersive visual experiences.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[2] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision (ECCV)*, 2016.

[3] G. Li, I. Yun, J. Kim, and J. Kim, "Dabnet: Depth-wise asymmetric bottleneck for real-time semantic segmentation," *arXiv preprint arXiv:1907.11357*, 2019.

[4] R. P. K. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: Fast semantic segmentation network," 2019.

[5] C. Castillo, S. De, X. Han, B. Singh, A. K. Yadav, and T. Goldstein, "Son of zorn's lemma: Targeted style transfer using instance-aware semantic segmentation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[6] L. Kurzman, D. Vazquez, and I. Laradji, "Class-based styling: Real-time localized style transfer with semantic segmentation," 2019.

[7] "Cityscapes dataset." `https://www.cityscapes-dataset.com/`. Accessed: June 15, 2023.

[8] V. Artists, "Wikiart dataset." `https://www.wikiart.org/`, Accessed: June 15, 2023.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "Pascal voc dataset 2012." `http://host.robots.ox.ac.uk/pascal/VOC/voc2012/`, 2012. Accessed: June 15, 2023.

[10] L. Sheng, Z. Lin, J. Shao, and X. Wang, "Avatar-net: Multi-scale zero-shot style transfer by feature decoration," 2018.

[11] Wikipedia contributors, "Composition vii." `https://en.wikipedia.org/wiki/Composition_VII`, Accessed: June 15, 2023.

[12] Wikipedia contributors, "Girl before a mirror." `https://en.wikipedia.org/wiki/Girl_before_a_Mirror`, Accessed: June 15, 2023.

[13] L. Afremov, "Mysterious rain princess." `https://afremov.com/mysterious-rain-princess.html`, Accessed: June 15, 2023.

[14] Wikipedia contributors, "The starry night." `https://en.wikipedia.org/wiki/The_Starry_Night`, Accessed: June 15, 2023.

[15] Wikipedia contributors, "The great wave off kanagawa." `https://en.wikipedia.org/wiki/The_Great_Wave_off_Kanagawa`, Accessed: June 15, 2023.

[16] Wikipedia contributors, "The persistence of memory." `https://en.wikipedia.org/wiki/The_Persistence_of_Memory`, Accessed: June 15, 2023.

[17] Wikipedia contributors, "Udnie." `https://en.wikipedia.org/wiki/Udnie`, Accessed: June 15, 2023.

[18] Wikimedia Commons, "Impression v (park) by wassily kandinsky, 1911." `https://commons.wikimedia.org/wiki/File:Impression_V_(Park)_by_Wassily_Kandinsky,_1911.jpg`, Accessed: June 15, 2023.

[19] Wikimedia Commons, "Vassily kandinsky, 1926 - several circles, guggenheim museum." `https://commons.wikimedia.org/wiki/File:Vassily_Kandinsky,_1926_-_Several_Circles,_Gugg_0910_25.jpg`, Accessed: June 15, 2023.

[20] Wikimedia Commons, "Upward by vasily kandinsky, 1929." `https://commons.wikimedia.org/wiki/File:Upward_by_Vasily_Kandinsky,_1929.jpg`, Accessed: June 15, 2023.

[21] Wikimedia Commons, "Kandinsky - jaune rouge bleu." `https://commons.wikimedia.org/wiki/File:Kandinsky_-_Jaune_Rouge_Bleu.jpg`, Accessed: June 15, 2023.