Critique of group 16 presentation - Style Transfer Using VGG

Critiques by group 3.

Overall: This project of Style Transfer clearly explains the purpose of style transfer to be applied in anime in the future. The explanations for the design of loss functions and the Gram matrix are clear. It explores one strategy, using a VGG network to train a style transfer model on famous Paintings.

1. The Dataset slide says, “it only contains dozens of pictures which is less than ImageNet.” The word “dozens” is confusing and vague. Make sure to specify a range of images like 50-100 or if it’s just an estimate, you can say it has fewer images than ImageNet per category.

2. The anime is called “Fate/Zero”, not “fate zero.”

3. How does the backup dataset get used in future style transfer work? Would explain it’s used for training future models for anime.

4. How did you create your own dataset? Was it from cropping images found online? Should mention that you resized all the images manually if you did so.

5. Would suggest explaining why use the Gram matrix versus other style representations? What are the benefits or drawbacks it has?

6. It’s better to place the training process slide before the results slide. That way, the presentation flows chronologically without a jump from the model to the results.

7. Training Process slide: Would be nice to see the loss function too though since it was explained at the beginning, but isn’t shown after that.

8. How many iterations is “Fully optimized”? Did you specify the model is fully optimized when you detected the lowest loss and stopped training there?

9. There was no conclusion about whether your model (VGG) worked well or how well it worked. Just a suggestion, but you could also explain how your model compares to other approaches like ResNet.

Reply to Group 3

1. Thanks for reminding us.

2. Thanks for reminding us.

3. We might use it if we need to train another network that can improve the performance of transferring into anime styles.

4. We collected by screenshot and the resized is done by simple code.

5. In fact we have subtracted the mean value from each channel in the image. Therefore, the function of the Gram matrix here is just the same as the covariance matrix. This is a standard way to calculate the correlation.

6. Thanks for your advice.

7. Yes, that would help remind the listeners what our loss function is.

8. In the slide, the full optimized iterations for The Great Wave style is 15k. We run for 15k iterations but it is not when we detect the lowest loss, after 5000 iterations, the loss remains in the same order of magnitudethat’s when we say the model is fully optimized.

9. Yes, the result can be hard to define, since it’s perceptual, the main metric used is loss, but whether the image is fully optimized is mainly based on perceptual quality. We did not mention the comparison in our presentation, but we compared our result with Gatys et al. in our report.

Critique of group 16 presentation - Style Transfer Using VGG

Critiques by group 8.

Group 12's project is about Neural Style transfer using VGG nets. Their presentation provides extensive discussion about the topic, thoroughly explaining the motivation, related works, method and results. The slide containing training results from different iterations is neat and Informative.

Some improvements/unclear

1. For the content and style representation, why choose those specific layers? Will the result be drastically different if you choose other layers?
2. Why use VGG net over other architectures e.g. ResNet? What is its main advantage on this task?
3. Is there a good metric for these kinds of tasks? Or is human evaluation the only one.
4. Does the style of the training dataset have an effect on the final model? Or are the results only dependent on the style of the input image (The Great Wave or The Starry Night).

Reply to Group 8

1. In our representation, we have talked about why we choose the output of the last convolution block to represent the content feature - because the last convolution block of a recognition model is invariant to a specific appearance, it is only sensitive to the content. For style representation, we select one output from each convolution block to make sure we catch the style totally. If you change the content representation to be far away from the output layer, the result could be drastically different, however, if you change the style feature layers, but you still keep selecting one output from each convolution block, you can get similar results.
2. The reason that we use VGG here is that it has wonderful performance in object recognition, and in the papers we read they all use VGG to get content and style features and calculate loss. Although ResNet can also be used to do object recognition, it has a different structure and we need to redefine the loss function to make sure it works fine, which is too difficult for us. The main advantage of using VGG is that we have a loss function defined by Gatys.
3. Loss is the main metric for neural style transfer tasks. We also use loss as our metric in the training process. But human evaluation is also crucial in style transfer tasks to determine whether the output loses too much content feature or lacks the style we want.
4. The training data set has nothing to do with the style image. The training dataset in our project is to train a transform network which gets the input from the dataset and output a rendered image. Then input the style image and the rendered output through the loss network, updating the weights of the transform network to make the rendered image more like the target we want.

Critique of group 16 presentation – Style Transfer Using VGG

Critiques by group 12.

Overall: This talk explained the background, literature survey, the dataset, feature extraction, optimization goal intuition, and the VGG19 model thoroughly. The method in the literature (mainly loss function) is successfully implemented and the artistic styles are applied correctly according to the result images shown. Using self-collected dataset is a perfect idea and makes this project more fascinating and I am really looking forward to seeing this team's future work.

Some improvements/unclear:

1. Except for the three styles mentioned in the Background part, you also tried the style of Pablo Picasso, which I believe should be added in the Background part. If there are many styles you tried and con not be fully exhibited in the background, you can state the number of styles tried and it will make the Background part more complete. 2. Is there any difference in the difficulty of applying these different styles? You can compare the speed of different styles reaches the same loss when applying those to the same content.

3. When applying the style transfer, if I catch on correctly, one picture of a style is used, so what's the reason for collecting a dataset of animation to train the anime style? Is there any other methods you are planning to use?

4. The size of self-collected dataset is not clearly given. Is there any classification in this dataset? (such as scenery and character) 5. Is there any reason for choosing VGG19 for this project? Have you tried other network structures or done some work in the comparison of structures?

6. How do you determine if the style of a picture is successfully transferred? Do you set a threshold of loss and how do you decide which value to choose?

Reply to Group 12

1. Thanks.

2. There is not much difference applying these different styles. The speed is almost the same, but the total losses are not in the same order of magnitude of the three styles.

3. In order to prove the generality of this style transfer network, we also create our own style dataset from our favorite animes. And we would like to try another method that improves the performance of anime style. This method requires more dataset.

4. most of them are daytime landscapes.

5. The reason that we choose VGG19 is that it has wonderful performance in object recognition. Sorry we don't try other network structures, we compare our results with Gatys' model in our project report.

6. We do not set the threshold of loss since loss varies greatly for different styles, they may not even be in the same order of magnitude. When the total loss stays steady in the same order of magnitude and never goes down (but there might be fluctuations), we determine the style is successfully transfered.

Style Transfer Using VGG (Group 16)

Xin Du
ECE Department
San Diego, US
xdu@ucsd.edu

Gaotong Wu
ECE Department
San Diego, US
g9wu@ucsd.edu

Zhuomin Zhang
ECE Department
San Diego, US
zhz056@ucsd.edu

***Abstract*—Style transfer is a widely used technique that can transfer real world images into a specific artistic style and keep the content of these real world images unchanged. In this project, we are going to implement a style transfer algorithm by using VGG19 as the feature extractor to transfer real-world images into our chosen various styles.**

I. Introduction

Style transfer is an optimization technique used to transfer a content image into the style of a style reference image (such as a work by a famous painter) - to make the output image resemble the content image. This is usually achieved by optimizing the output image to match the content feature representations of the content image and the style feature representations of the style reference image.These features can be extracted from the image using pre-trained network on image classification task, such as VGG19.

In art area, there are many different styles created by different artists, such as Van Gogh's the Starry Night, which belongs to post-imrepssionism, The Great Wave which is the traditional Japanese style, and we also have the Scream belonging to the expressionism. We will apply these artistic styles for our own images making them resemble the corresponding style.

II. Related Work

Many people have tried to solve this problem by proposing different techniques. Efros and Freeman [1] use an image quilting algorithm to do texture transfer in 2001. It is given an input image, produces an output image that are both visually similar to and pixel-wise different from the input, and having possibly a larger size. The algorithm consists of two parts: analysis and synthesis, where analysis denotes estimating a set of relevant statistics from the input texture image and synthesis denotes computing an image that satisfies the statistical constraints estimated during the analysis step.

Fujun Luan and Sylvain Paris [2] propose Deep Photo Style Transfer It runs several pipeline steps for computing the final transfer image, creates a segmentation mask, groups segmentation classes, defines and pre-computes loss functions and gradually optimizing the transfer image;

We also have the most famous work is done by Leon A. Gatys [3], Alexander S.Ecker and Matthias Bethge in 2016, which uses deep learning network to extract features and content from the input image, train a capable network to generate a specific style. Our project mainly uses this as reference,

III. Dataset and Features

A. Dataset

The dataset we use is ImageNet1000(mini) from Kaggle. It is a subset of ImageNet. It has 1000 categories of images, just the same as ImageNet. However, in each category, it contains only dozens of images. This dataset contains the natural images to train the model. Fig.1 shows some of the images from our dataset.

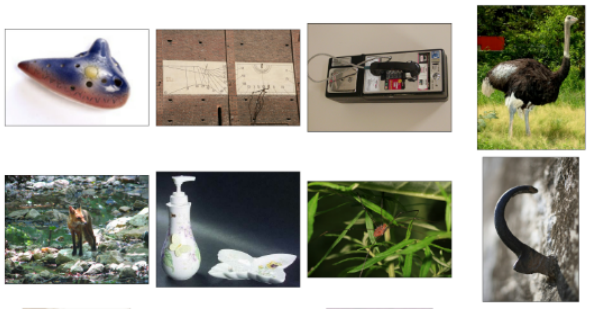

Fig. 1. Natural images from ImageNet1000(mini)

We also collect several style images from Google. For example, the masterpiece from Pablo Picasso – "The muse", a Japanese painting called "Under the Wave off Kanagawa". We want to transfer the style of a natural image to the style of one of these style images. Fig.2 [8] [9] shows some of these style images.

Fig. 2. Style image examples

B. Feature Representation

1) Content Representation: Consider a CNN trained on object recognition. If the network is robust enough, then the output should be sensitive to the content of the image, no matter what the precise appearance is, the CNN should always

recognize the content of the image [3]. Therefore, we can regard the features generated by the convolution layers near the output layer as our content representation.

2) Style Representation: The way we use to define style features of an image is Gram matrix. [4] Gram matrix calculates inner products of every two features in the feature space. If we reshape the features to be vectors, and we denote the feature space $V = \{v_1, v_2, ..., v_n\}$, then Gram matrix G will be a $n \times n$ matrix, whose entries are given by (1).

$$G_{ij} =< v_i, v_j > \tag{1}$$

The reason to use Gram matrix to represent style features is that, Gram matrix calculate the differences between two features. If two features are same, then the entry of Gram matrix is large, if two features are totally different, then the corresponding entry of Gram matrix is closed to zero. If the result image and the style image have a similar style, then the features of the two images in each layer should have the similar relation.

IV. Methods

A. System Overview

The system we use consists of two parts, a transform network to generate the result image, and a loss network to calculate the loss function [5]. The system is shown in Fig.3. Our goal is to tune the weights in the transform network to minimize the loss function.

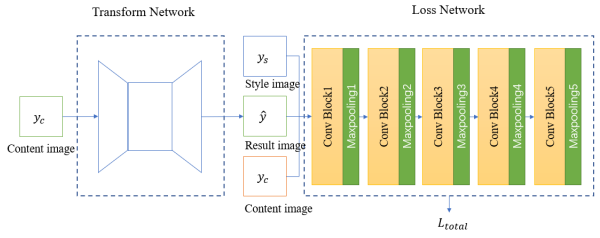

Fig. 3. System overview

B. Image Transform Network: ResNet

The transform network we use is a residual network [5]. The reason for using residual network here is that it can prevent the vanishing gradient problem [6], and therefore, it has better performance than CNN in learning features. The ResNet structure is shown in Fig.4.

C. Loss network: VGG-19 Model

The model we use is basically VGG-19. VGG is a convolutional neutral network which has very good performance on image recognition. Fig.5 [10] shows the architecture of VGG-19. The input image size should be $(256 \times 256 \times 3)$, and the output is an (1000×1) vector to represent which category the image belongs to. We use the pretrained VGG-19 model without fully-connected layers to be our loss network. We obtain content and style features from this model, then use these features to calculate the loss function, and tune weights in the transform network to minimize the loss function.

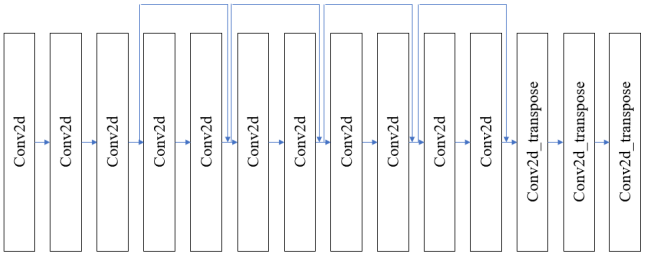

Fig. 4. Transform Network

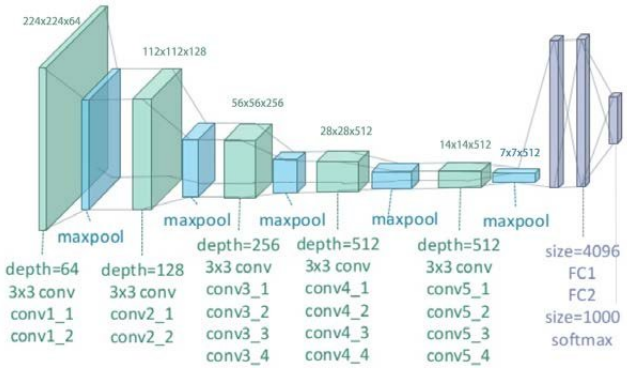

Fig. 5. VGG-19 architecture

D. Loss Function

1) Content Loss: Let $\vec{x}$ be the original natural image, and $\vec{y}$ be the result image we generate. A convolution layer with N_l distinct filters has N_l feature maps. We resize the feature maps to be vectors, each vector has the same dimension M_l. We denote the the feature space of original image in layer l as X^l, and denote the feature space of result image in layer l as Y^l. Therefore, $X^l, Y^l \in R^{N_l \times M_l}$, X^l_{ij} denotes the output of original image of filter i at position j in layer l. Then the content loss can be defined by the squared error loss between two content representations

$$L_{content}(\vec{x}, \vec{y}, l) = \frac{1}{2} \sum_{i,j} (Y^l_{ij} - X^l_{ij})^2 \tag{2}$$

2) Style Loss: Let $\vec{s}$ be the style image, and S^l be the feature space of style image in layer l. We denotes the Gram matrix of style image and result image in layer l as G^l and A^l respectively. We use the other notations defined in the content loss. Then the style loss of one layer is calculated by:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{ij} - A^l_{ij})^2 \tag{3}$$

Therefore, the total style loss is calculated by:

$$L_{style} = \sum_l (\omega_l E_l) \tag{4}$$

where ω_l is the weight for layer l.

3) Total Loss: The total loss is defined by the weighted sum of the above three loss functions.

$$L_{total} = \alpha L_{content} + \beta L_{style} \tag{5}$$

α and β are hyperparameters here. For different style image, we need to assign different values to these parameters manually to make the style transformation successful.

E. The Whole Algorithm: Neural Style Transfer

For each iteration, the algorithm processes the following items:

- Resize each image in the batch to be $(256 \times 256 \times 3)$. Subtract the mean value from each channel.
- Feed the image batch to the transform network to get the output images.
- Feed the output images to loss network(VGG-19), according to the content layer and style layers we select, calculate the loss function.
- Use Adam optimizer to tune the weights in the transform network, in order to optimize the loss function.

V. Experiments and Results

The goal of style transfer is to generate an image which combines the content of an image y_c with style of a style image y_s. For each style, e.g., The Great Wave by Hokusai, in the training process the parameters are fixed for the loss network, we only update weights in image transform network. After the training, we get the checkpoint file for this style so that through restoring checkpoint we can get a quick output.

A. Training Details

We train our neural style transfer model on the ImageNet1000(mini) dataset. The dataset contains 38669 training images, We resize each of the image to 256×256 and train with a batch size of 4 for 15k iterations, giving roughly two epochs over the training data. We use Adam optimizer with learning rate 1×10^{-3}. We do not use weight decay or dropout since the model does not overfit within two epochs. Our training runs on Google Colaboratory with GPU for almost 4 hours per style. Fig.8 shows outputs of The Great Wave style from certain iterations.

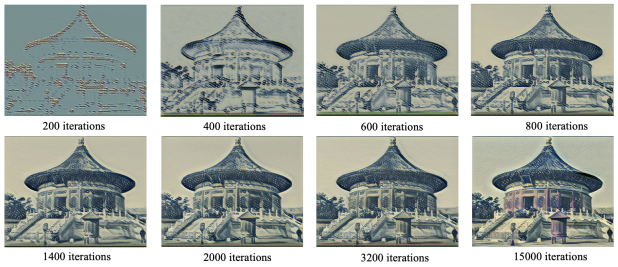

Fig. 6. Outputs from certain iterations.

The total loss function in equation (5) is the metric we use to train our transform network. Fig.**??** show the loss in our training process for The Great Wave style. The total loss is as big as 10 million, the loss stays at the same order of magnitude after 5000 iterations, although with fluctuations, we stop at 11500 iterations and that is when we call it is fully optimized. The content weight=7 and style weight=100 in this training process, which means the output will be more like the content.

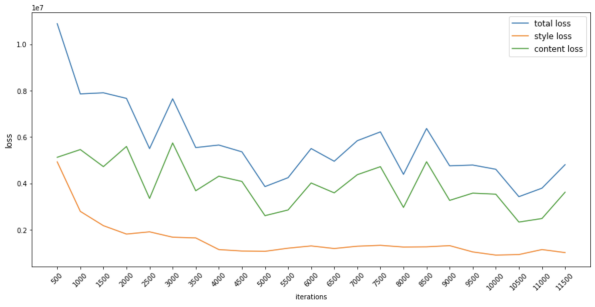

Fig. 7. Total loss, content loss and style loss every 500 iterations for The Great Wave style training.

Before training a style, we test the content loss weight and style loss weight for Gatys's method which has faster training process than ours. We choose the proper weight for each style. We also tried out different content representations for the model, relu4 and relu5 presents similar results since they are the nearest layer to the output. We did not try out different style representations since we all the layers to represent style.

B. Results

Although our model are trained given the input resized to 256×256, it can output images with all sizes. In Fig.8 [11]we show examples of style transfer using our models on images with different size.

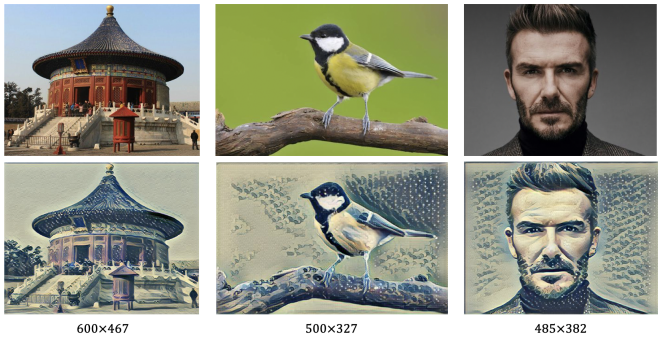

Fig. 8. Example results for style transfer on multiple sizes of images by applying models trained on 256×256 images. The style image is The Great Wave. The image on the left with size 600×467, the image in the middle with size 500×327, the image on the right with size 485×382.

Our results are highly perceptual, in Fig.8 we show qualitative examples comparing results with Gatys *et al.* [3] In all three cases the hyperparameters α, β in equation (5) are exactly the same between the two methods. Overall, in Fig.9 [12] the results from two methods are similar, but our method preserves more features from the content image, also with

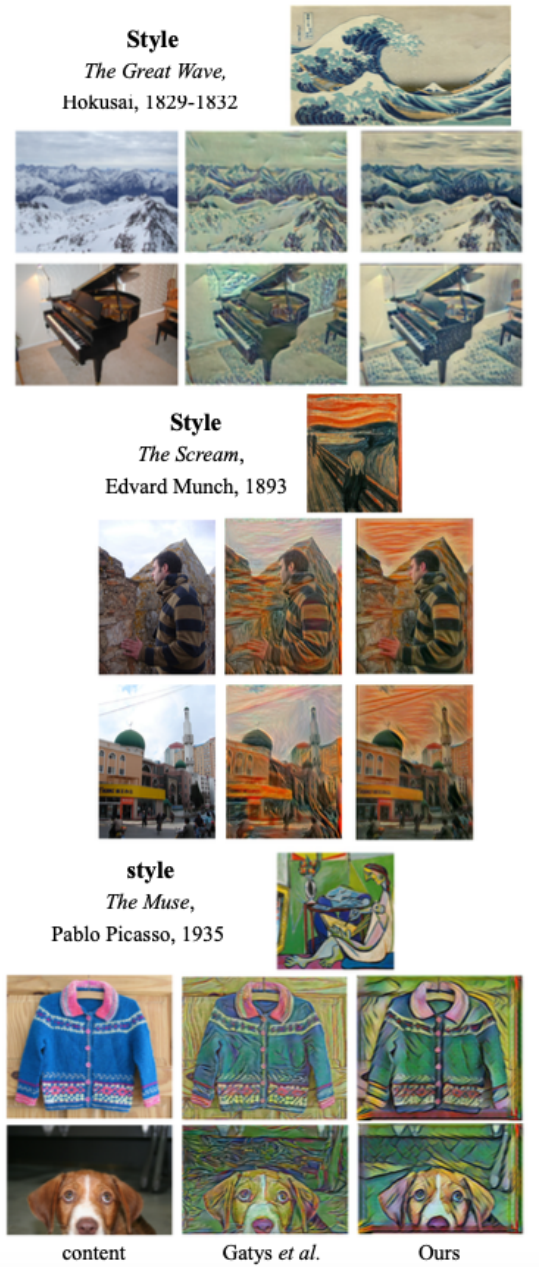

Fig. 9. Example results of the three styles we trained from our style transfer network compared to Gatys *et al*. The hyperparameters α, β are exactly the same between the two methods. For The Great Wave style on the top, $content \quad loss : style \quad loss = 7 : 100$, for The Scream style in the middle, $content \quad loss : style \quad loss = 25 : 100$, for The Muse style on the bottom, $content \quad loss : style \quad loss = 16 : 100$.

greater perceptual quality, e.g., in the style of The Scream, the architecture generated by Gatys is distorted and is not in the same shape as the content image. In the style of The Great Wave, the mountain generated by Gatys lacks details of the edge shape of the mountain while ours remains the edge perfectly. Also, the cloth and dog images in The Muse style from our method are closer to the style image in color.

C. Discussion

The results are highly qualitative for distinctive artistic style, however the main issue of the project would be using style transfer on other style such as anime, In Fig.10 we show the result of anime style we trained, but we did not achieve satisfactory animation results. Our method usually have some problems with anime, among which significant problems mainly include: 1) the generated images have no obvious animated style textures; 2) the generated images lose the content of the original images; 3) the parameters of the network require the large memory capacity.

Fig. 10. Left: Original. Middle: Anime Style. Right: Result of anime style.

Generative Adversarial Networks (GAN) can also solve the style transfer problem. CartoonGAN by Chen *et al*. [7] transforms photos of real-world scenes into cartoon style images. Training on high-quality dataset can generate animation images with better visual quality than our mothod.

Fig. 11. Result of anime style with GAN.

VI. Conclusion

In this paper, we have applied neural networks that can transfer real world images into a specific artistic style and keep the content of these real world images unchanged. In future work we hope to explore more artistic style, e.g., anime, and the use for other image transformation tasks.

VII. Contribution

Xin Du: (1)Preprocess and visualize the dataset; (2)Develop the learning model; (3)Train the model with different hyperparameters; (4)Write part IV and V in the report; (5)Write some reply to peer review.

Gaotong Wu: (1) Collect and resize training dataset for anime style; (2) Test and build up model for anime style transfer, try GAN;(3) Write abstract, introduction, related work and part V in the report; (5) write some reply to peer review;

Zhuomin Zhang: (1)Train three style models; (2)Improve the model by trying out different hyperparameters (3)Compare results with other style transfer methods; (4)Write Experiments, Results, Discussion and Conclusion part; (5)Reply to reviews regarding training and results.

References

[1] Lara Raad, and Bruno Galerne, Efros and Freeman Image Quilting Algorithm for Texture Synthesis, Image Processing On Line, 7 (2017), pp. 1–22. https://doi.org/10.5201/ipol.2017.171

[2] Luan, Fujun et al. "Deep Photo Style Transfer." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017): 6997-7005.

[3] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2414-2423. 2016.

[4] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Texture synthesis using convolutional neural networks." arXiv preprint arXiv:1505.07376 (2015).

[5] Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." In European conference on computer vision, pp. 694-711. Springer, Cham, 2016.

[6] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.

[7] Chen, Yang, Yu-Kun Lai, and Yong-Jin Liu. "Cartoongan: Generative adversarial networks for photo cartoonization." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 9465-9474. 2018.

[8] Pablo Picasso. "A Muse". Available: https://www.wikiart.org/en/pablo-picasso/a-muse-1935

[9] Katsushika Hokusai. "Under the Wave off Kanagawa". Available: https://www.metmuseum.org/art/collection/search/45434

[10] https://www.researchgate.net/figure/llustration-of-the-network-architecture-of-VGG-19-model-conv-means-convolution-FC-means_fig2_325137356

[11] Temple of Heaven image available: https://images.app.goo.gl/CzbRyhV78zQjTF468; Bird image available: https://images.app.goo.gl/uZgRbZpTUzhH4TAG7; Beckham image available: https://images.app.goo.gl/wmS5Lvp8Hyk6J6ZD8.

[12] All the content images are from ImageNet1000 dataset. Available: https://www.kaggle.com/ifigotin/imagenetmini-1000.

[13] Figure 10 Original. Available:https: //imgur.com/r/KamenRider/Et3Mj1i

[14] Figure 10 anime style. Available:https://theculturetrip.com/asia/japan/articles/places-in-tokyo-that-inspired-makoto-shinkai/