Portrait Stylization Based on Deep Convolutional Neural Network

Guangrui Ding [*] **Tom Panenko** [†] **Zhenghao Sun**[‡] **Shangzhou Yin**[§]

1 Task

Image style transfer has been widely used in many industrial applications, such as cartoons, videos, and Adobe Photoshop, and aims to recompose images in the style of other images and thus transform them into vivid color pastiches.[1] We seek to apply artistic styles to portrait images with newly developed Deep convolutional neural networks. Previous studies introduce VGG-16 as a popular deep learning technique for image style transfer with the advantage of simplicity and fast training speed, so we implement it as the base model. However, the VGG-16 network cannot perform well for images with complex and detailed features such as portraits(i.e. wrinkles, hair and etc.), in order to keep more specific features for human faces, we adopt a novel DualStyleGAN that provides a more natural way of style transfer by defining the content and style of an image with an intrinsic style path and a new extrinsic style path, which enables to control both color and structural styles hierarchically to precisely pastiche the style image[2]. The input of models is a content image that is passed to the model and a style image that is preferred over the content image. The output styled image is recomposed of content and style images.

2 Related Work

In 2016, Justin Johnson et al. from Stanford University introduces a fast stylization method that improves the approach proposed by Gatys, which can generate stylized images in just a few seconds on the GPU. The network structure adds an image transformation network, essentially involving and connecting the last deep residual convolutional network (Resnet) with a loss network using a pre-trained VGG-16[3] network. The loss value from the model and the parameters in the image transformation network are continuously updated and optimized until the final ideal model is obtained. Luan et al. describes a deep learning approach to handle diverse of image content while suppressing distortion[4]. Even though this approach is able to handle the diversity of scenes, it is not a suitable method to perform style transfer for portraits due to limited ability to show details of human faces.

Due to the rise of generative adversarial networks(GAN) and its ability to generate high-quality images, researchers find GAN to be a suitable technique for image style transfer. StyleGAN network is more advanced than the GAN network and can handle video generation with better textural detail recovery[5].

In addition to this, Exemplar-Based High-Resolution Portrait Style Transfer[2] improved on StyleGAN with DualStyleGAN, a portrait generation style transfer model that separates images into intrinsic and extrinsic style paths. By doing so, the model more accurately separates the relevant element of the images that need to be updated in the transfer. Along with this, the model is able to make changes to intrinsic characteristics, extrinsic characteristics, or both, supporting changes to both the color profiles and the content of the original image.

3 Approach

We use the VGG-16 network as the base model to recompose the content of an portrait in the style of another. To extract more detailed features and maintain color and structural styles for portraits, we explore and implement a more suitable model specifically designed for enhancing image quality of portrait style transfer.

3.1 VGG-16 network

A key characteristic of training CNN networks is that the corresponding receptive field in the feature

[*] Dept. of ECE, Boston University grding@bu.edu
[†] Dept. of ECE, Boston University tompan@bu.edu
[‡] Dept. of ECE, Boston University szh1007@bu.edu
[§] Dept. of ECE, Boston University syin10@bu.edu

map of the lower layers which close to the input layer is relatively small and thus results that more texture and detailed information can be extracted. As the number of layers of network increases, the corresponding receptive field in the feature map closing to the output layers becomes larger and contains more abstract features, referring to style information of the target image.

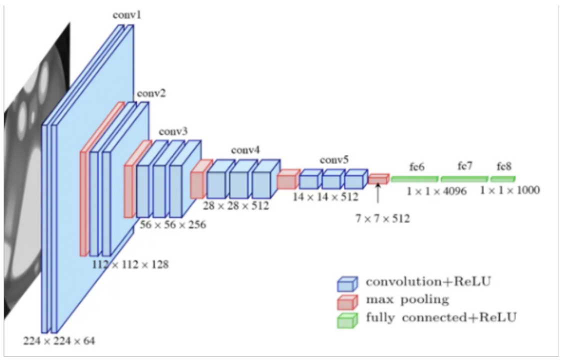


Figure 1: The architecture of the VGG-16 network

Based on the above property, after passing the style image into the model, we obtain the feature maps containing most of the detailed information from the first five convolutional layers of the VGG-16 network. The content image is passed into the model and its feature map is selected and obtained from the fifth and sixth convolutional layers, as well as its content loss function.

3.1.1 Gram matrix

The essential aspect of applying the VGG-16 network for image stylization is to determine how to obtain the feature map of each layer, with the critical step of acquiring these feature maps being the application of the Gram matrix.

The Gram Matrix is an eccentric covariance matrix, and covariance can often be considered as a type of second-order statistic. Therefore, using the Gram Matrix can be known as a method for calculating the degree of cross-correlation between different channels of the stylized output feature map. The resulting inner product between each channel's feature map can be obtained, so the outcome after the Gram Matrix operation should be a C*C matrix. This matrix is capable of effectively capturing global statistical data.

First, assume that the feature map of the input style image I_s at layer l is: $\mathcal{F}^l(I_s) \in R^{C\times H\times W}$, where C represents the number of channels, and H and W respectively denote the height and width of $\mathcal{F}^l(I_s)$. If we perform a transformation on $\mathcal{F}^l(I_s)$, such that $\mathcal{F}^l(I_s)' \in \mathcal{R}^{C\times(HW)}$, after processing with the Gram Matrix, we can obtain:

$$\mathcal{G}(\mathcal{F}^l(I_s)') = \left[\mathcal{F}^l(I_s)'\right]\left[\mathcal{F}^l(I_s)'\right]^T \in \mathcal{R}^{C\times C} \tag{1}$$

We can see that when calculating the Gram matrix for three-dimensional vectors, it involves taking the dot product of the i-th channel and the j-th channel and then summing them. Through the inner product operation, regions with larger values in the feature map will become even larger due to the inner product computation. This effectively scales the image's characteristics, highlighting the features and essentially capturing the style of the image.

3.1.2 Style loss function

The style loss function serves as a measure of whether the generated image has achieved the given style standard. The derivation process for the style loss function is as follows: By combining the previously discussed VGG-16 network structure and Gram matrix, we can obtain the feature maps stemmed from the Gram matrix operation for each layer in the network structure. We then define the style loss function as the squared Euclidean distance between the Gram Matrix representations of the output result image and the provided style image. Based on this, we can derive the formula:

$$\mathcal{L}_s = \sum_{l\in\{l_s\}} ||\mathcal{G}(\mathcal{F}^l(I_s)') - \mathcal{G}(\mathcal{F}^l(I)'||^2 \tag{2}$$

Here, l_s represents the set of all layers in the VGG-16 network used for calculating the style loss function. $\mathcal{L}_s$ is the style loss function we are seeking. The following section is the discussion of obtaining the content loss function.

3.1.3 Content loss function

Similar to the style loss function, we obtain the feature maps induced from the Gram matrix operation at the selected convolutional layers in the VGG-16 network. We define the content loss function $\mathcal{L}_c$ as the squared Euclidean distance between the Gram Matrix representations of the output result image and the provided content image at the l layer in the VGG network. Based on this, we can derive the formula:

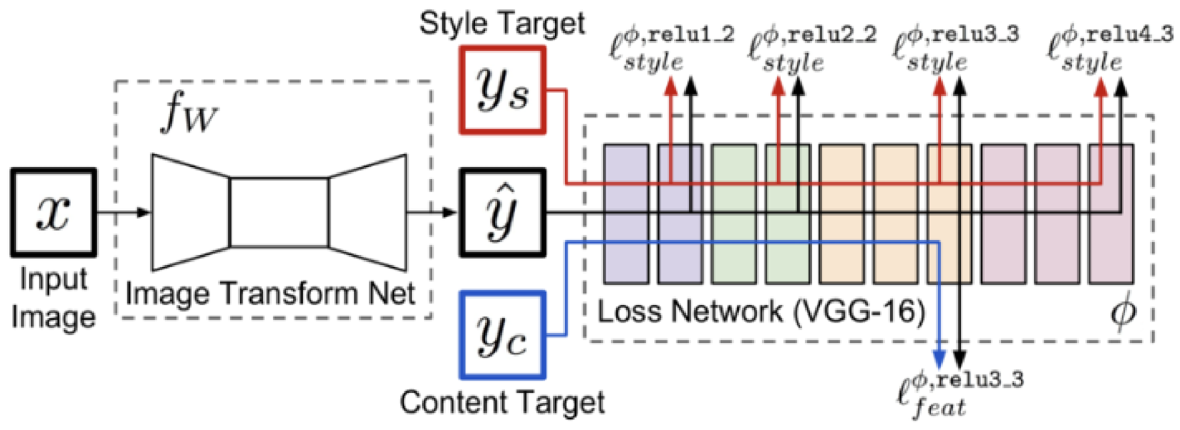

Style Target
y_s
f_W
x
Input Image
Image Transform Net
$\hat{y}$
y_c
Content Target
Loss Network (VGG-16)
ϕ
$\ell_{style}^{\phi,\mathtt{relu1\_2}}$
$\ell_{style}^{\phi,\mathtt{relu2\_2}}$
$\ell_{style}^{\phi,\mathtt{relu3\_3}}$
$\ell_{style}^{\phi,\mathtt{relu4\_3}}$
$\ell_{feat}^{\phi,\mathtt{relu3\_3}}$

Figure 2: Training process graph

$$\mathcal{L}_c = \sum_{l \in \{l_s\}} ||\mathcal{F}^l(I_c)' - \mathcal{F}^l(I)'||^2 \tag{3}$$

Then we combine these two loss function to get the total loss function.

3.1.4 Total loss function

The total loss function for image stylization is defined simply, as a linear combination of the style loss function and the content loss function. However, we assign different weights to the two loss function components, usually giving a larger weight to the style loss function to emphasize style transfer. The expression for the total loss function is as follows:

$$\mathcal{L}_{\text{total}}(I_c, I_s, I) = \alpha\mathcal{L}_c(I_c, I) + \beta\mathcal{L}_s(I_s, I) \tag{4}$$

In this equation, I_c represents the input content image, and I_s represents the input style image. $\mathcal{L}_c$ and $\mathcal{L}_s$ denote the content loss function and style loss function defined earlier. α and β are the weight parameters used to balance the content and style elements, where α represents the weight parameter for the content loss function and β represents the weight parameter for the style loss function. To obtain a satisfying and desired stylized image, we need to minimize the system's loss function value, which leads to:

$$\begin{aligned} I^* &= \arg\min_I \mathcal{L}_{\text{total}}(I_c, I_s, I) \\ &= \arg\min_I \alpha\mathcal{L}_c(I_c, I) + \beta\mathcal{L}_s(I_s, I) \end{aligned} \tag{5}$$

After iterating the well-designed network, the ideal effect is to make I^* gradually decrease to reach the minimum value.

3.1.5 Backpropagation

During the continuous iterative process, the gradient values from backpropagation are used to update the data of the to-be-output image, making it closer to the style of the style image and more similar to the content of the content image. (The to-be-output image is initialized as random noise before the network operates). We define the gradient update formula for the content loss function as:

$$\frac{\partial \mathcal{L}_c}{\partial \mathcal{F}^l(I_c)'} \tag{6}$$

We define the gradient update expression for the style loss function as:

$$\frac{\partial \mathcal{L}_s}{\partial \mathcal{F}^l(I_s)'} \tag{7}$$

After updating the gradients, the updated results are back-propagated to the output image. As the output image undergoes continuous updates and iterations, its content becomes increasingly similar to the content image, and its style progressively approaches the style image. Concurrently, the total loss function diminishes in value.

3.2 DualStyleGAN Portraits Stylization

For the purpose of comparison, we plan to use the DualStyleGAN network for stylization transfer in portraits. This novel architecture can effectively realize the modeling and control of dual styles for the examplar-based style transfer, by retaining an intrinsic style path of styleGAN to control the style of the original domain, while adding the extrinsic style path to model and control of the target domain. The implementation of this method comprises four main parts.

3.2.1 Facial Destylization

The first step is facial destylization, which aims to recover realistic faces from artistic portraits to create anchored face-portrait pairs for supervision. Destylization allows for a multi-stage approach to balance facial realism and portrait fidelity, involving latent initialization, latent optimization, and image embedding.

3.2.2 DualStyleGAN

Once the dataset is preprocessed, we can input the data to the DualStyleGAN architecture, which combines GAN networks and residual blocks. The network architecture is shown below.

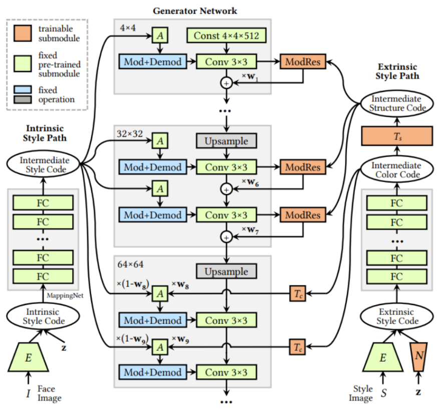


Figure 3: DualstyleGAN network structure

Besides the intrinsic and extrinsic path, the main generator network consists of 18 layers. For the coarse resolution layers (1-7), modulative residual blocks $ModRes$ are proposed to adjust structure styles and add a structure transform block T_s to characterize domain-specific structural styles. For the fine resolution tuning part (8-18) the extrinsic style path takes the same strategy as StyleGAN. This approach provides an efficient way to achieve hierarchical modeling, flexible style manipulation, model collapse, and structure preservation.

3.2.3 Progressive Fine-tuning

The next step is to apply progressive fine-tuning to smoothly transform the StyleGAN result toward the target domain. This process consists of three stages that handle color, structure transformation of the source domain, and style transformation of the target domain, respectively. The loss function can also be defined within, which can be devided into 4 parts (8).

$$\min_G \max_D \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{perc}}\mathcal{L}_{\text{perc}} + \mathcal{L}_{\text{sty}} + \mathcal{L}_{\text{con}} \quad (8)$$

where the first term is StyleGAN adversarial loss. By decreasing hyperparameters, the extrinsic path is able to capture and transfer more structure styles rather than the color. The second term is the perceptual loss, which is introduced to improve the resolution. Perceptual loss measure high-level perceptual and semantic differences between images. Style and content loss are also included, which is shown in equation (9) and (10).

$$\begin{aligned}\mathcal{L}_{\text{sty}} = \lambda_{\text{CX}}\mathcal{L}_{\text{CX}}(G(\mathbf{z}, \mathbf{z}_e^+, \mathbf{1}), S) + \\ \lambda_{\text{FM}}\mathcal{L}_{\text{FM}}(G(\mathbf{z}, \mathbf{z}_e^+, \mathbf{1}), S)\end{aligned} \quad (9)$$

$$\begin{aligned}\mathcal{L}_{\text{con}} = \lambda_{\text{ID}}\mathcal{L}_{\text{ID}}(G(\mathbf{z}, \mathbf{z}_e^+, \mathbf{1}), g(\mathbf{z})) + \\ \lambda_{\text{reg}}||W||_2\end{aligned} \quad (10)$$

3.2.4 Latent Optimization and Sampling

We perform latent optimization and sampling in the last step. It is hard to fully capture the extremely diverse styles. To solve this problem, we fix DualStyleGAN and optimize each extrinsic style code to fit its ground truth S. The optimization follows the process of embedding an image into the latent space and minimizes a perceptual loss and a contextual loss in Equation (9) and (10).To sample random extrinsic styles, we train a sampling network N to map unit Gaussian noises to the distribution of optimized extrinsic style codes using a maximum likelihood criterion. we treat these two parts separately, i.e., structure code and color code are independently sampled from N and concatenated to form the complete extrinsic style code.

3.2.5 Implementation details

Although we are not able to trained the Network by ourselves, we still provided the implementation details here. The progressive fine-tuning uses eight NVIDIA Tesla V100 GPUs and a batch size of 4 per GPU. For the structure transfer on the source domain, the $\lambda_{adv} = 0.1, \lambda_{perc} = 0.5$ and trained on 300, 300, 3000 iterations, which takes about 0.5 hour. For the style transfer on the target domain, the $\lambda_{adv} = 1, \lambda_{perc} = 1, \lambda_{CX} = 0.25, \lambda_{FM} = 0.25$, set $(\lambda_{ID}, \lambda)_{reg})$ to $(1, 0.015), (4, 0.005), (1, 0.02)$ and trains for 1400, 1000, 2100 respectively. Training takes about 0.75 hour. Destylization, latent optimization and training sampling network use one GPU and take about 5, 1, 0.13 hours, respectively. Testing takes about 0.13s per image.

4 Datasets

We test the VGG-16 and DualStyleGAN on the Toonify Cartoon dataset. For training purpose, 317 figures should be applied. For further comparison, we use 15 more images to test the SSIM and PSNR, which can evaluate the content loss transfered results.

5 Evaluation Metrics

Besides the loss function mentioned in the VGG network and DualStyleGAN, to achieve parallel comparison, structural similarity index measure (SSIM), equation (11), which is a method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos and peak signal-to-noise ratio (PSNR), equation (12), which focused on the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation are also applied.

$$\mathrm{SSIM}(x,y)=\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \tag{11}$$

$$\begin{gathered}\mathrm{MSE}=\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)-K(i,j)]^2\\ \mathrm{PSNR}=10\cdot\log_{10}\left(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}}\right)\end{gathered} \tag{12}$$

6 Results

6.1 VGG-16 stylization performance

Now, we will show the results generated by VGG-16 network with different hyperparameters.

6.2 Portrait stylization

Then we focused on the protrait stylization tasks. **VGG-16 Performance** We implemented the sylization result on the VGG nework, seeing fig7. From result, only obvious detail manipulation can be observed, i.e. color of the face and background.

DualStyleGAN Performance We also implemented DualStyleGAN, seeing fig 8. Many latent features are changed in the target images, i.e. shape of eyes, curve of the face and shadow of the nose. The right part also demonstrate the flexible way to choose the appropriate result. Another amazing function is the style fusion, which is shown in fig9. DualStyleGAN also provides a method to merge

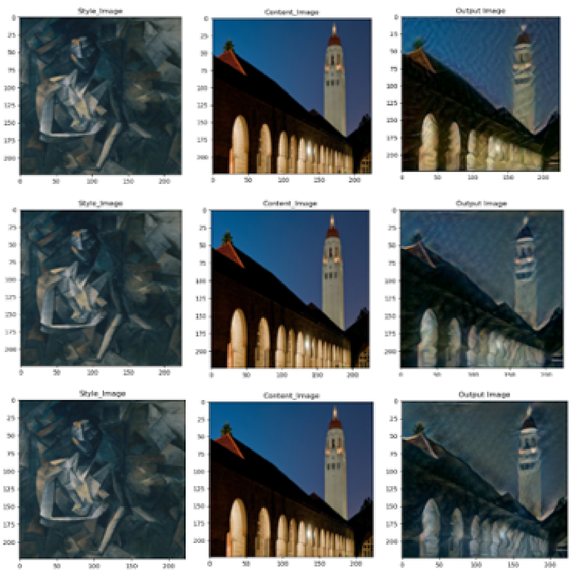


Figure 4: An example of outputs of style transfer with 5000 in style weight but 50, 100, 500 in epoch.

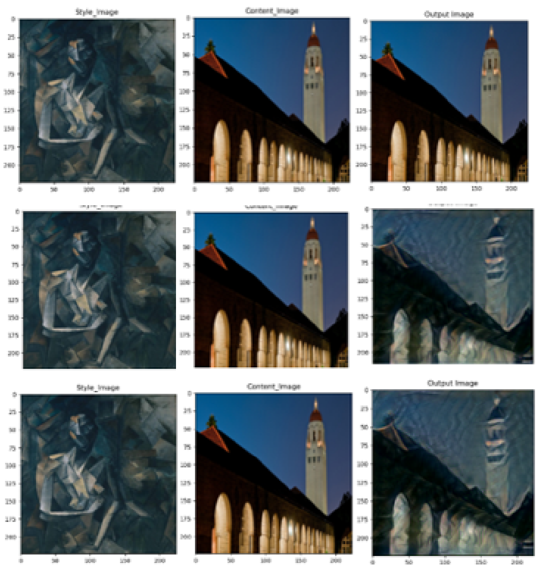


Figure 5: An example of outputs of style transfer with 200 in epochs but 10,100, 5000 in style weight.The figure shows the different epochs.

multiple styles together to transfer on the content image and the weight between different styles can also be modulated.

Parallel comparison We applied SSIM and PSNR to measure the content similarity and loss of the output, seeing fig9. No matter SSIM or PSNR validate that DualStyleGAN has a better performance than VGG-16. For completeness, user study should be conducted. However, we have no time to do so.

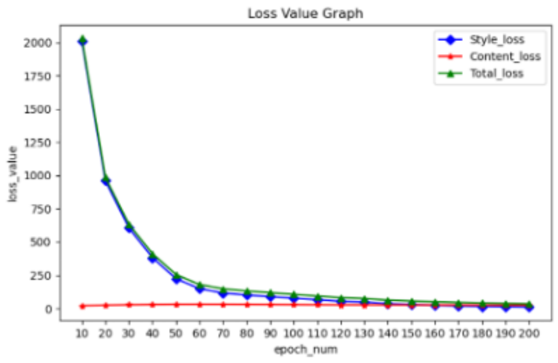


Figure 6: An example of loss graph with 200 in epochs and 100 in style weight.

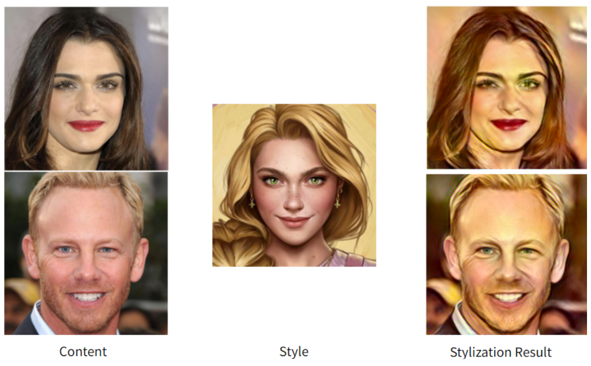
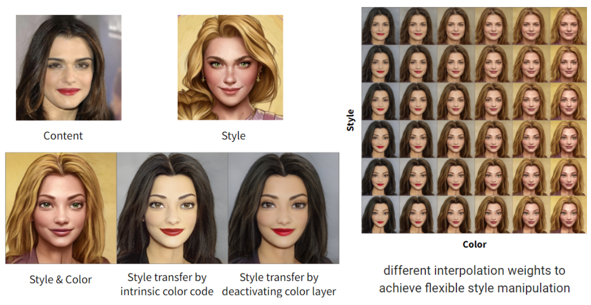


Figure 7: Stylized portraits images generated using VGG-16 network.



Content
Style
Style
Color
Style & Color
Style transfer by intrinsic color code
Style transfer by deactivating color layer
different interpolation weights to achieve flexible style manipulation

Figure 8: Stylized portrait images generated using DualStyleGAN network.

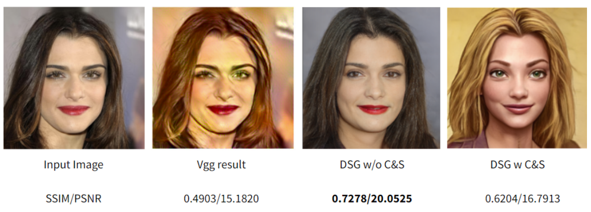


Figure 9: Comparison of VGG-16 and DualStyleGAN results.

7 Conclusion

Compared to the VGG-16, DualStyleGAN provides the art of state stylization methods to achieve high resolution, training efficiency, exemplar based color and style tranformation and a more flexible way to choose the appropriate result. DualStyleGAN might also be able to applied on the video stylization. Biomedical image virtual staining may share potential insights with style transfer, which means DualStyleGAN might be helpful.

References

[1] Manjeet Singh. Artistic style transfer with convolutional neural network. https://medium.com/data-science-group-iitr/artistic-style-transfer-with-convolutional-neural-network-7ce2476039fd, 2017.

[2] Shuai Yang, Liming Jiang, Ziwei Liu, and Chen Change Loy. Pastiche master: exemplar-based high-resolution portrait style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7693–7702, 2022.

[3] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016.

[4] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4990–4998, 2017.

[5] Tianyi Wei, Dongdong Chen, Wenbo Zhou, Jing Liao, Weiming Zhang, Lu Yuan, Gang Hua, and Nenghai Yu. E2style: Improve the efficiency and effectiveness of stylegan inversion. *IEEE Transactions on Image Processing*, 31:3267–3280, 2022.

A Detailed Roles

This section is shown in Table 1.

Name	Task	File Name	No. Lines of Code
Guangrui Ding	Implemented DualStyleGAN, show result and calculate SSIM/PSNR	dualstylegan-VGG-quantification.m, Readme.md	300+
Tom Panenko	Organized LaTeX format Wrote all formula, import references	Readme.md, Vgg-16.py	100+
Zhenghao Sun	Implemented VGG-16 and show result	StyleTransfer-VGG16.py, Vgg-16.py, Readme.md	600+
Shangzhou Yin	Wrote Section 1 and 2	Readme.md, Vgg-16.py	300+

Table 1: Team member contributions

B Code Repository

The Github Repository link is as follows: https://github.com/Sun-zhenghao-BU/Deep-Learning-Project--EC523