Ludwig-Maximilians-Universitt Mnchen
Major: Computer science

Pytorch Neural Style Transfer With CLIP

Machine Vision and Learning - Master Practical

Chenke Xie
Matrikelnummer: 12346871

Supervisors:
PROF. DR. BJOERN OMMER
DR. Matthias Wright

Munich, 08/2022

Abstract

Neural style transfer has shown exciting results, enabling new forms of image processing. However, rendering the semantic content of images in different styles is a complex image processing task. Most of the previous style transfer algorithms relied on a non-parametric approaches for texture synthesis and preserved the structure of the target image in a different way[1][2][3]. Arguably, a major limiting factor of previous approaches is the lack of image representations that explicitly represent semantic information. Leon A. Gatys proposed a neural algorithm[4] that can separate and recombine the image content and style of natural images. This algorithm is able to extract the semantic image content from the target image and then inform a texture transfer procedure to render the semantic content of the target image in the style of the source image. It achieved remarkable result. Since Convolutional Neural Networks can be used to extract high level image information explicit, the algorithm used VGG network[5] to derive image representations. From this point, I thought of CLIP model[6], which has various advantages, such as zero-shot prediction, flexibility, etc. I tried to replace VGG network with CLIP model, in order to analyze the effect of different computer vision model on this algorithm.

Table of Contents

1. Introduction

1.1 CLIP

State-of-the-art computer vision systems are trained to predict a fixed set of predetermined object categories. This restricted form of supervision limits their generality and usability since additional labeled data is needed to specify any other visual concept. Learning directly from the raw text about images is a promising alternative that leverages a much broader source of supervision.

The development of text-to-text as a standardized input-output interface has enabled task-agnostic architectures to zero-shot transfer to downstream datasets removing the need for specialized output heads or dataset-specific customization. These results show that modern pre-training methods in web-scale collections of text can obtain overall supervision that exceeds that obtained with high-quality crowd-labeled NLP datasets.

So paper[6] proposed a model called CLIP, Contrastive Language-Image Pre-training, with 400 million pairs of graphical datasets from the network, using text as image labels. To perform the downstream task, only the text description corresponding to the concepts on the graph needs to be provided and zero-shot transfer can be performed.

The model experimented on 30 CV datasets with experimental tasks including OCR, action recognition in videos, Geo-localization, and many types of fine-grained object classification. The model is optimal for most of the tasks. Moreover, it is generally comparable to other baseline models without the need for specific training.

1.2 Neural Style Transfer

Transferring the style from one image onto another can be considered a problem of texture transfer. In texture transfer, the goal is to synthesize a texture from a source image while constraining the texture synthesis in order to preserve the semantic content of a target image. For texture synthesis there exist a large range of powerful non-parametric algorithms that can synthesize photorealistic natural textures by resampling the pixels of a given source texture. Most previous texture transfer algorithms rely on these non-parametric methods for texture synthesis while using different ways to preserve the structure of the target image. Although these algorithms achieve remarkable results, they all suffer from the same fundamental limitation: they use only low-level image features of the target image to inform the texture transfer.

Ideally, however, a style transfer algorithm should be able to extract the semantic image content from the target image (e.g. the objects and the general scenery) and then inform a texture transfer procedure to render the semantic content of the target image in the style of the source image. Therefore, a fundamental prerequisite is to find image representations that independently model variations in the semantic image content and the style in which it is presented. Such factorized representations were previously achieved only for controlled subsets of natural images such as faces under different illumination conditions and characters in different font styles or handwritten digits and house numbers.

To generally separate content from style in natural images is still an extremely difficult problem. However, the recent advance in Deep Convolutional Neural Networks has produced powerful computer vision systems that learn to extract high-level semantic information from natural images. It was shown that Convolutional Neural Networks trained with sufficient labeled data on specific tasks such as object recognition learn to extract high-level image content in generic feature representations that generalize across datasets and even to other visual information processing tasks, including texture recognition and artistic style classification.

In the original paper[4], they showed, how the generic feature representations learned by high-performing Convolutional Neural Networks form VGG can be used to independently process and manipulate the content and the style of natural images. At the same time, it introduces *A Neural*

Algorithm of Artistic Style, a new algorithm to perform image style transfer. Conceptually, it is a texture transfer algorithm that constrains a texture synthesis method by feature representations from state-of-the-art Convolutional Neural Networks. Since the texture model is also based on deep image representations, the style transfer method elegantly reduces to an optimization problem within a single neural network. New images are generated by performing a pre-image search to match feature representations of example images. This general approach has been used before in the context of texture synthesis and to improve the understanding of deep image representations.

1.3 CLIP and Neural Style Transfer

CLIP is proposed to learn visual concepts based on linguistic supervision, which is trained on unfiltered, highly disparate, and highly noisy 400 million data from the Internet almost without human annotations. And it learns a wide range of visual concepts directly from natural language. It has shown strong Zero-shot performance in benchmark dataset tests such as ImageNet classification. Therefore, I believed it has great potential for style transformation tasks as well. However, it is very difficult to apply CLIP as a Zero-shot model directly to the style transformation task, because the CLIP task is complex multi-modal inference. Therefore, I proposed to integrate CLIP with the existing style transformation model by replacing its original convolutional neural network with the vision portion of CLIP.

2. Method

A Neural Algorithm of Artistic Style algorithm[4] proposed a model based on convolutional neural networks to perform image style transfer. The core idea of the model is to use deep convolutional neural network to extract features of images and build the content representation of the content image and the style representation of the style image. The two feature representations are used to synthesize a target image that matches both content representation and style representation.

2.1 VGG

The original model used a normalized version of the 16 convolutional and 5 pooling layers of the 19-layer VGG network[5] to extract image features. The authors defined this VGG network and modified the forward function, in order to return the feature values $F^l \in R^{(N_l \times M_l)}$ for each layer, where N_l is the channels of layer l and M_l is the height of the feature map multiplied by its width. All feature values are stored in a list.

With the list of feature values, the authors defined the content representation and the style representation by comparing the feature values. Here **Mean-Squared-Error(MSE)** loss is adopted to calculate the loss. For each layer of the convolutional neural network, the feature representation in layer l of the target image $\vec{p}$ is denoted as P^l, and the feature representation in layer l of the content image $\vec{x}$ is denoted as F^l. Then the loss function can be defined as

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F^l_{i,j} - P^l_{i,j})^2 \tag{2.1}$$

The style representation is the main contribution of this algorithm since it has been very tricky to extract its style representation from a style image. The model considers that the style representation of an image can be built from the top filters of any layer of the convolutional neural network. The model represents style representations as the correlations between the different filter responses. The authors built a matrix called **Gram Matrix**, denoted as $G^l \in R^{(N_l \times N_l)}$, where $G^l_{i,j}$ is the inner product between the vectorised feature maps i and j in layer l.

$$G^l_{i,j} = \sum_k F^l_{ik} F^l_{jk} \tag{2.2}$$

Let $\vec{a}$ be the input style image, $\vec{x}$ be the target image, and their style representations in layer l are denoted as A^l and G^l. Then the style loss in layer l is denoted as

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{i,j} - A^l_{i,j})^2 \tag{2.3}$$

The authors also set different weighting factors w_l to the total loss between different layers, in order to set the contribution of the layer to the total loss. The total style loss is obtained by multiplying the loss and the weighting factors between the layers and then accumulating them.

After obtaining the content loss and style loss, the authors defined a total loss function

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x}) \tag{2.4}$$

for the target image. Using the gradient descent method and **L-BFGS** optimizer to optimize each pixel of the target image. So the model can synthesize a target image that matches both the image representation and the style representation.

2.2 CLIP-RN50

I know from the model that extracting the feature maps of the image using a convolutional neural network[7] is an essential part of building the image representations. So, the performance of the convolutional neural network determines the performance of the model. Start from that idea, I proposed to use another network model of computer vision replace VGG network.

I thought of using the CLIP model to replace the VGG model. CLIP is a powerful multi-modal learning model with strong transfer learning capability and amazing ability in image classification problems. CLIP contains many models and I first used RN50 to perform it.

Because CLIP is a multi-modal learning model, the model contains two portions, one of which is the computer vision model and the other is the natural language processing model. I extracted the visual portion called **ModifiedResNet50** from RN50 model and discarded the natural language processing portion. I used ModifiedResNet50 to extract image features. For convenience, RN50 is used below instead of ModifiedResNet50.

RN50[8] is a deep convolutional neural network. Compared with VGG, it has deeper layers and uses residual block. I believed that low-level features and high-level features can be better represented in deep convolutional neural networks. I also expect the model to have higher performance.

RN50 uses multiple residual network blocks. Since I do not have access to the weight parameters of the CLIP model, I cannot modify the forward function of RN50 to return the feature values of different layers as VGG does. For this purpose, two approaches are used to obtain the feature values in RN50. The first approach is to define a function called *vision forward* to implement the feed forward passing of images in RN50. Each layer of RN50 is extracted and recombined into a sequential module. The images are passed into the sequential module and the return values of each layer are stored. The second approach is to define the *forward hook* function and register the forward hook for the required layer in RN50, then when the image is passed into RN50, the forward function of RN50 will automatically trigger the forward hook to return the output of the required layer. A comparison of the results of the two approaches is shown in the section 3. RN50 contains multiple residual network blocks, and each block of the residual network contains multiple bottleneck blocks. I imagined each bottleneck output analogously as a single layer 2D convolution output in VGG and store all outputs in a list. Since feature maps are available, I could define the loss function in the same way. The weighting of the loss layers I adjusted according to the channels of the different layers of RN50.

2.3 CLIP-ViT-16/B

The impressive performance of *Transformer*[9] in natural language processing has also brought new ideas to the field of computer vision, so the *Vision Transformer(ViT)*[10] was born. ViT model has also excellent performance in image classification problems without using convolutional neural network. After using CLIP's RN50 model, I would like to see if I could use CLIP's ViT-16/B model to extract the image information instead of the traditional convolutional neural network. ViT-16/B contains a visual portion and a natural language processing portion like RN50. I only need the visual portion of the ViT-16/B model. ViT-16/B uses one layer of convolutional layers to split a 224*224 image into 196 patches of 16*16 patch size, each of which is 768 dimensions. In addition to the 196 image tokens, a class token is concatenated and a positional embedding is added, so the image becomes a two-dimensional tensor with 197*768 shape. I used the **forward hook** method to extract the two-dimensional tensor from the ResidualAttentionblocks of ViT. For the ViT model, it is not possible to extract the feature maps like convolutional neural networks, so I cannot calculate the **gram matrix** for the feature values as I did before. I imaged that the first dimension of the tensor associated with the recognition and localisation of the image, which represents the correlation between different pixels. From this point, I modified the definition of the gram matrix accordingly,

$$G = F \cdot F.transpose(1, 2) \tag{2.5}$$

where F is input three-dimensional tensor with batch size, patch numbers and embedding dimensions.

3. Evaluation

I will evaluate the performance of the model in two dimensions: processing time and quality of the synthesized image.

3.1 RN50

3.1.1 forward-vision version

The environment I used is google Colab pro. The GPU is Tesla P100(16GB). I first evaluated the forward-vision version. The performance is impressive, this model takes only half a minute to synthesize a target image under 500 iterations(I tested 200, 500, 1000, 1500 iterations and found that the loss decreases very slowly after 500 iterations, so 500 iterations are enough). With the same training resources and environment, the VGG model takes 1.5 hours to synthesize a target map. This shows that CLIP's RN50 has very strong performance, almost 180 times faster than the VGG model.

I compared the results in terms of the synthesized images. The result is shown in 3.2. With the VGG model, the style pattern is more obvious on the target image, while the style pattern on the generated image using the RN50 model is relatively small. This means that RN50 does not compute the style representation very well. I also evaluated different layers to compare the contributions of different layers for style extraction. But unfortunately the results are similar and there is no significant improvement. Figure 3.1 shows the synthesized images using different layers to compute the style representation.

3.1.2 forward-hook version

After searching the CLIP source code, I concluded that the model using the forward vision approach does not take advantages of the RN50 residual block. So I proposed to use the forward hook function, which can extract the feature values of the required layers without changing the RN50 forwarding. This model using the forward hook still maintains excellent performance. I tried to train the model with GPU, but for some unknown reason, the pixel gradient computed under GPU will be NAN. So I trained the model under CPU. The CPU is Intel(R) Xeon(R) CPU @ 2.20GHz.

Even with CPU only, the model takes only 5 minutes to synthesize a target image, which is 18 times faster than the VGG model. From Figure 3.2 I can conclude that it is obvious that the model using forward hook function synthesizes images with more obvious style pattern. However, neither is good enough compared to the VGG model. I used different layers to calculate the style representation and content representation and compared the results. The results are shown in

Figure 3.1: I set the content layer to *(Layer 1 bottleneck 0) layer* unchanged and show the results by comparing different style layers. From left to right, the first image uses *(relu1, relu2, relu3, Layer 1 bottleneck 0, Layer 1 bottleneck 1)* as the style layer. The second one uses *(relu1, relu2, relu3, Layer 1 bottleneck 0, Layer 2 bottleneck 0)* as style layers. The style layers of the third sheet are *(relu3, Layer 1 bottleneck 0, Layer 1 bottleneck 1, Layer 2 bottleneck 0, Layer 2 bottleneck 0)*. The style layers of fourth image are *(relu3, Layer 1 bottleneck 0, Layer 2 bottleneck 0, Layer 3 bottleneck 0, Layer 4 bottleneck 0)*. The style layer of the fifth one is *(Layer 1 bottleneck 0, Layer 1 bottleneck 1, Layer 2 bottleneck 0, Layer 3 bottleneck 0, Layer 4 bottleneck 0)*.

Figure 3.2: Style Transfer results using three different model. The right image is content image. I chose three style images for test. From left to right, they are: *Femme nue assise by Pablo Picasso, The Starry Night by Vincent van Gogh, Composition VII by Wassily Kandinsky*

Figure 3.3: I set the content layer to *(Layer 1 bottleneck 0) layer* unchanged and show the results by comparing different style layers. From left to right, the first image uses *(relu1, relu2, relu3, Layer 1 bottleneck 0, Layer 1 bottleneck 1)* as the style layer. The second one uses *(relu1, relu2, relu3, Layer 1 bottleneck 0, Layer 2 bottleneck 0)* as style layers. The style layers of the third sheet are *(relu3, Layer 1 bottleneck 0, Layer 1 bottleneck 1, Layer 2 bottleneck 0, Layer 2 bottleneck 0)*. The style layers of fourth image are *(relu3, Layer 1 bottleneck 0, Layer 2 bottleneck 0, Layer 3 bottleneck 0, Layer 4 bottleneck 0)*. The style layer of the fifth one is *(Layer 1 bottleneck 0, Layer 1 bottleneck 1, Layer 2 bottleneck 0, Layer 3 bottleneck 0, Layer 4 bottleneck 0)*. It can be seen that the deep layers cannot compute the style representation well.

Figure 3.4: I set *(relu1, relu2, relu3, Layer 1 bottleneck 0, Layer 1 bottleneck 1)* as style layers and show the results by comparing different content layers. From left to right, the first image uses *Layer 1 bottleneck 0* as content layer. The second one uses *relu1*. The third one uses *relu3*. The fourth one uses *Layer 2 bottleneck 0*, and the last one uses *Layer 3 bottleneck 0*. It shows that the most content information will be lost in deep layers.

Figure 3.3 and 3.4. It can be concluded that the convolutional neural network is better than bottleneck network in pixel information extraction. Since the layers of RN50 network are very deep, if the content layer is chosen too deep, then the most content information will be lost and cannot be reconstructed. Similarly, if when the style layers are chosen too deep, the style information cannot be well represented. Therefore, after evaluation, I selected the style layers as *(relu1, relu2, relu3, layer1 bottleneck 0, layer1 bottleneck 1)* in RN50. For the content layer, I selected *layer1 bottleneck 0*.

3.2 ViT-16/B

ViT-16/B model does not use convolutional neural network for image processing, I defined a new gram matrix so that ViT can compute the style representation. I outputted the result of the gram matrix and found that the values are too small to calculate the loss. so I removed the div function, in order to make the loss larger. However, after 500 iterations, the result image does not achieve style transfer. The picture 3.5 shows the training phrase. It is possible that the definition of the gram matrix is not accurate. But the ViT model also maintains the same high performance and generates the target graph within 10 minutes. So after getting the exact definition of style representation in ViT model, the model is still very promising.

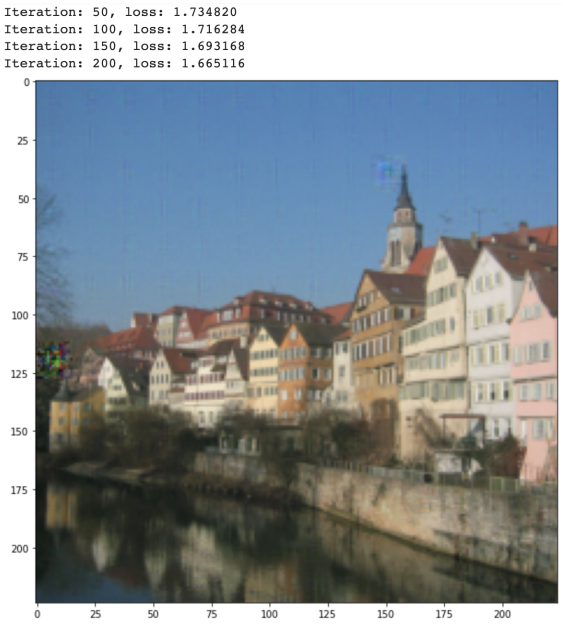

Figure 3.5: Synthesized image using ViT

3.3 Feature Maps Visualization

I visualized the feature maps of content image and style image for the VGG model which in the original paper, Clip-RN50 with hook and Clip-RN50 forward-vision.

The purpose of visualization of feature maps is to get a more intuitive picture of how the image changes after going to a particular layer. It is possible to visualize the changes after a particular layer and to compare the previous layer to get the changes in each layer. The model is no longer a "black box" inside.

At the same time, visualization of three different models can be compared after a specific number of layers, which can be used to help determine the differences between the two models and better compare algorithms.

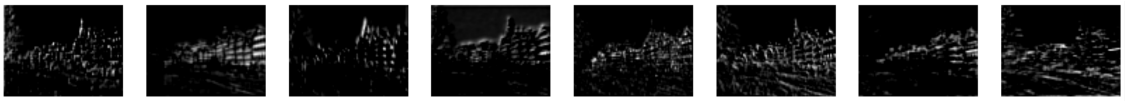

Figure 3.6: Feature map of the content image after VGG model($relu(self.conv4_2(out['r41']))$)

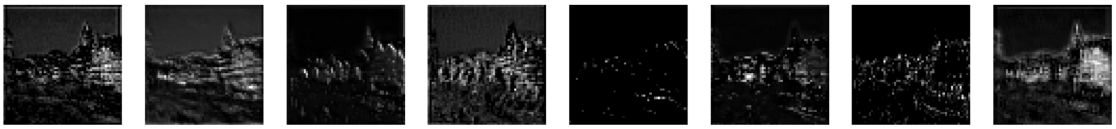

Figure 3.7: Feature map of the content image after Clip RN50 model with hook(*layer1 bottleneck 1*)

Figure 3.8: Feature map of the content image after Clip RN50 model without hook(*layer1 bottleneck 1*)

The images above are part of the results of the content images after passing through each of the three content layers.

From the visualized feature map, I can know that after passing Clip-RN50 forward-vision layer, two images are no longer visible and no semantic image content can be extracted from them.

In addition, the performance of the VGG model and Clip-RN50 hook-vision is similar.

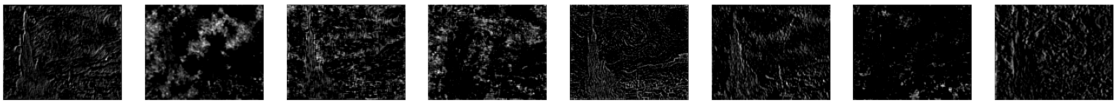

Figure 3.9: Feature map of the style image after VGG model layer2(*(relu(self.conv3*$_1$*(out*$['p2']$*)))*)

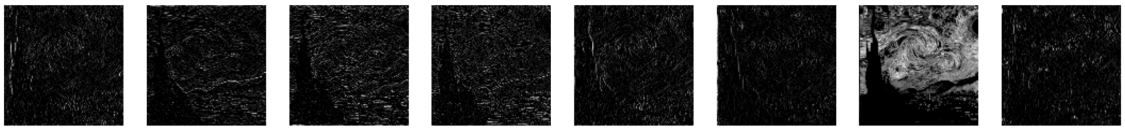

Figure 3.10: Feature map of the style image after Clip RN50 model with hook(*layer1 bottleneck 1*)

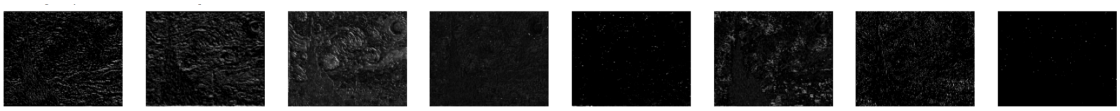

Figure 3.11: Feature map of the style image after Clip RN50 model without hook(*layer1 bottleneck 1*)

The above images are some of the results of the style images after passing through their third layer respectively.

From the visualized feature map, I can see that after passing through the Clip-RN50 forward-hook layer, one of the textures of the style image can be very clearly preserved.

In addition, the performance of the VGG model and Clip-RN50 forward-vision is basically similar.

In summary, the content map and style map can be obtained by the corresponding layer of feature maps.

By comparison, I can learn that the content map obtained by Clip RN50 forward-vision has the least semantic image content, and it can be inferred that the final result should be worse, while VGG and Clip RN50 hook-vision cannot see obvious differences.

4. Discussion and Conclusion

4.1 Discussion

I started out by getting the layers directly from CLIP RN50's model and recording the feature map after passing the images through. Then some of the layers are directly used as parameters to pass into the loss function to train the content map and style map. The result is that the training time is greatly reduced with CLIP RN50 forward-version, from an hour and a half for the VGG model to about 0.5 minutes. However, CLIP RN50 forward-version does not transfer the texture extracted from the style map to the content map as well as the VGG model.

Later I learned that the RN50 model is a residual network. The residual network is used to skip connections or shortcuts to skip some layers. the RN model uses a double or triple layer jump (ReLU) containing non-linearity and a batch normalization in between. There are two main reasons for adding jump connections: to avoid the problem of gradient disappearance, thus leading to easier to optimize neural networks, where the gating mechanisms facilitate information flow across many layers ("information highways"), or to mitigate the Degradation (accuracy saturation) problem; where adding more layers to a suitably deep model leads to higher training error. During training, the weights adapt to mute the upstream layer, and amplify the previously-skipped layer.

Due to the above residual network, I cannot obtain a layer directly from the RN model, which will result in the difference between the obtained layer and the actual layer parameters, so I used "hook" to obtain the corresponding layer from the RN50 model to solve the problem of residual network, so that the individual layer taken out is consistent with the model.

Compared to the VGG model, the model with both methods of CLIP RN50 greatly reduces the training time of the model. From about an hour and a half, it is reduced to about 0.5 or 5 minutes. CLIP RN50 using forward-hook shows better texture than CLIP RN50 using forward-Vision, but still not as good as VGG's. However, CLIP RN50 using forward-hook greatly reduces the training time compared to the VGG model. Therefore, compared to the other two CLIP RN50 using forward-hook is a more suitable compromise solution, which takes into account the training time and also obtains better texture effect in the style image.

How to define the style representation when using ViT model is a tricky problem. Since in a convolutional neural network, the feature maps are three-dimensional tensor, which contain semantic content and can be visualized. But in ViT, the output of the intermediate encoder layer is always a two-dimensional tensor. It represents a input image split into 196 patches, each of which is mapped to 768 dimensions after linear embedding. The semantic content information of input image contained inside this tensor after multiple layers of encoder is not known. I tried to compute the gram matrix of the style image by using 196 patches as channels, but I found the total loss will be very small. In order to fix it, I deleted div function. Unfortunately, the result shows that there is still no style representation. But the result still showed the performance of the ViT model. Using ViT for style transfer is promising, if the correct definition of the loss function can be found.

4.2 Conclusion

In summary, I replaced the VGG model with the CLIP model in Neural Style Transfer. The style layer texture extraction is implemented and added to the content layer. After the model is completed, I adjust the number of extracted layers according to the actual situation and try to find the combination with the best effect. Finally, when I compared the Neural Style Transfer With CLIP model with the original model, I concluded that the training time of the model is greatly reduced while achieving good texture transfer effect. It is a more suitable choice in terms of performance.

Bibliography

[1] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001. i

[2] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001. i

[3] N Ashikhmin. Fast texture transfer. *IEEE computer Graphics and Applications*, 23(4):38–43, 2003. i

[4] Leon A Gatys, 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*, pages 2414–2423, 2016. i, 1, 3

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. i, 3

[6] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. i, 1

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[8] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. 2015. 4

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 4