Generating Artistic Images Using Neural Style Transfer with Pre-Trained Convolutional Neural Networks

Nguyen Duc Tri *, Phuong T. Tran †
* *Faculty of Electrical and Electronics Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam*
† *Wireless Communications Research Group, Faculty of Electrical and Electronics Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam*
Emails: *41900297@student.tdtu.edu.vn, tranthanhphuong@tdtu.edu.vn*

***Abstract*— In fine arts, especially painting, people have mastered the skill of creating unique visual experience through creating a complex interweaving of content and style way of an image. In important areas of visual perception, for example object and face recognition, near-human performance has recently been demonstrated by a class of biologically inspired vision models called Deep Neural Networks. Here, we apply an artificial system based on Deep Neural Network, which helps reate artistic images of high perceptual quality. The system uses neural models to decompose and recombine the content and style of arbitrary images, provides a neural algorithm to generate artistic images.**

Keywords*— *artistic image, neural style transfer, convolutional neural networks, content loss, style loss

I. INTRODUCTION

Neural Style Transfer (NST) is a technique in computer vision that takes two images — a content image (Content Image) and a style reference image (Style Image) — and mix them together so that the output image retains its characteristic elements content image, but appears to be "drawn" in the style of the reference style image [1-3]. For example, a photo of a cat would provide visual content and a van Gogh painting would be Style reference image. The output will be a self-portrait looking resembles an original photo by van Gogh.

Fig. 1. Example of Neural Style Transfer

The image on the left is called the content image and the image on the side must be called a style image. We are interested in stylization one image (left image in this case) using another image (image right). This is what makes up the last two words in the term - style transfer). To perform this process, the neural network is trained to optimize the loss function custom loss function, hence the first word neuron. When the above two images are merged using neural style transfer, the final output looks like the figure right.

Neural Style Transfer (NST) is one of the most interesting techniques in deep learning (Deep learning). It is an algorithm that creates a new image starting from a content image (the cat in the image above) and a style image. The goal is for the generated image to contain the "content" of the content image, but have the same "style" as the style image. NST is a form of transfer learning using a Pretrained Convolutional Neural Network (CNN) [4].

The new and interesting point in this problem is: Instead of training the weights of the network, the author trains the generated image, which is used as input to the network, until the goal is fully achieved. The algorithm was created by Gatys et al. [5]. The author implemented NST in TensorFlow using a pretrained VGG-19 network pre-trained on the ImageNet dataset [6].

In Neural Style Transfer, we are interested in stylizing a real world image (content image) using another style image (style image). To do this, the author builds a model based on a neural network that is trained to optimize the loss function. When the two images are merged using the model designed by the author, the final output is the stylized image like the image below.

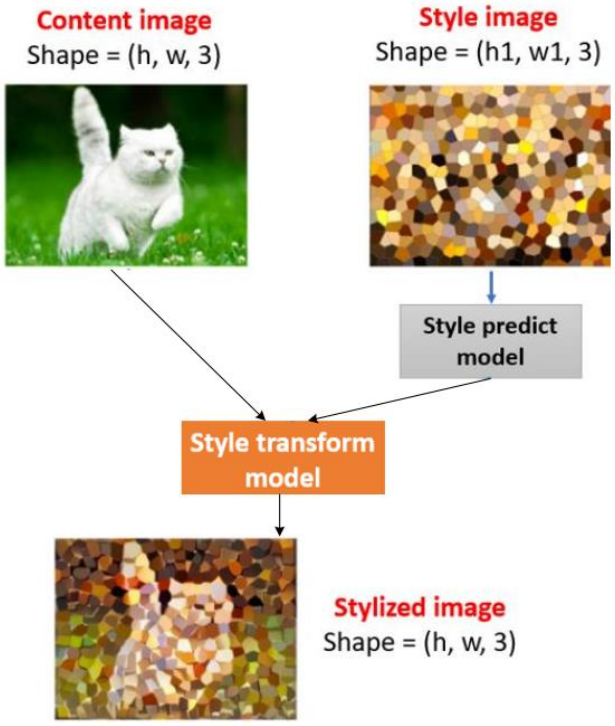

Fig. 2. Idea of the algorithm

- Content Image after going through Style transform model will retain the features and spatial structure of the original image.

- Style Image after going through Style predict model will have its spatial structure broken, continuing through Style transform model will get the outstanding features of Style Image.

- Stylied Image is the image that has been stylized after combining 2 input images, Content Image and Style Image.

The main contributions of this work includes:

- Design a Style transform model and Style predict model, then synthesize them.

- More specifically, the main task in this lesson is that we will design 2 functions: Content Loss and Style Loss.

To realize the problem, we need to create an additional noise image with pixel values that can be changed, it acts as a variable, automatically updating the new state after each training, so we call it Noise Image.

• Noise Image: After each training, the Noise Image will gradually change to carry the features and spatial structure of the Content Image, and the style is similar to the Style Image.

• Target 1: In reality, Target 1 is the Content Image, after going through the Pretrained Model it will retain the features and spatial structure of Target 1.

• Target 2: In reality, Target 2 is the Style Image, after going through the Pretrained Model it will retain the features of Target 2, but the spatial structure will be changed.

Our goal is to separate the features of Content Image and Style Image, from the perspective of Pretrained Model, it will detect the prominent features of the image. And here, to suit the problem, we use Pretrained VGG-19 network.

The organization of this paper is as below: the next section we will introduce the background of neural networks. In Section 3, we perform experiments on two datasets using our proposed methods. Section 4 concludes the paper.

II. BACKGROUND

A. *Supervised Learning Approaches To The Image Classification Problem*

The branch of machine learning that we turn to in order to solve the problem of image classification is called supervised learning. In fact, when we hear most people talking about machine learning today (deep learning or otherwise), what they are probably referring to is supervised learning, as it is the subset of machine learning that has proven most successful in recent years [7]. Supervised learning is now the classic procedure for learning from data, and it is outlined below in the context of the image classification problem:

(1) We start with a set of pre-classified example images, which means we have a set of images with known labels. This is called the training data, and these data serve as the ground truths that our system will learn from.

(2) The function we are trying to find is called the score function, which maps a given image to a category score. To determine what we are looking for, we first guess its functional form and make it depend on a series of parameters $\boldsymbol{\theta}$ that we need to find.

(3) We have a loss function, $\boldsymbol{L}$, that quantifies the discrepancy between what our scoring function suggests for the item score and what our training data provides as the known truth. This loss function is therefore increased if the scoring function performs poorly and decreased if it performs well.

(4) And finally, we have a learning or optimization algorithm. This is a mechanism for feeding our system a set of training data and letting it iteratively improve the scoring function by adjusting its parameters $\boldsymbol{\theta}$. The goal of the learning process is to identify the parameters that give us the best loss.

(5) Once we have completed this process and learned a suitable scoring function, we hope that it will generalize well. That is, the function performs well for general inputs (called test data) that it has never seen before.

B. *Neural Networks*

To make the scoring function nonlinear, we use a neuron. This is a simple function, shown in the following animation, that does three things: (1) it multiplies each of its inputs by a weight; (2) it sums these weighted inputs into a single number and adds a bias; (3) It passes this number through a nonlinear function called an activation and produces an output.

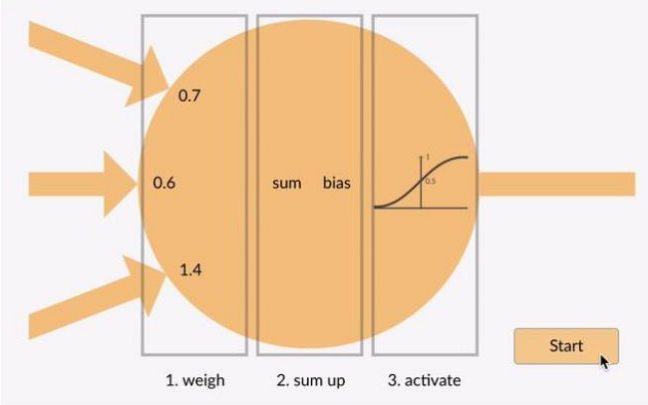

Fig. 3. An artificial neuron (a perceptron).

These neurons can be arranged into layers to form a neural network where the outer layers match the shape of our inputs and outputs. For the MNIST dataset, this is a vector of 784 incoming numbers and 10 outgoing numbers. The middle layer is called the hidden layer because we do not have direct access to it on the input or output. It can be of any size, and this is what defines the architecture of the network. In neural network parlance, the network shown below is called a two-layer network or a one-hidden-layer network. (We can have as many of these hidden layers as we need).

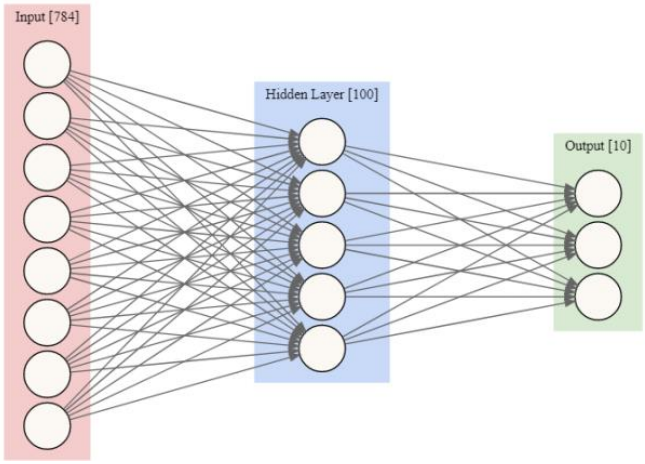

Fig. 4. An artificial neuron (a perceptron).

C. Convolutional Neural Networks

There are some fundamental drawbacks to the general form of neural networks we have seen so far (the term for them is standard or fully-connected neural networks). First, they completely ignore the 2D structure of the image to begin with. Instead of working with a 28 × 28 matrix as input, they work with a 784-digit array as input. And we can imagine that some useful information about the neighborhoods that share pixels is being lost. Second, the number of parameters we need to learn grows rapidly as we add more layers.

Convolutional neural networks (CNN) were built to solve both of these problems, making them particularly powerful for processing image data. CNN are very similar to fully connected neural networks, but what makes CNN special is that they make explicit assumptions about the structure of the input, in our case the image, allowing them to encode certain good properties in their architecture. These choices make them efficient to implement and significantly reduce the number of parameters in the network.

Instead of processing the input data (and arranging the intermediate neural layers) as a linear array, they process the information as a 3D volume (with width, height, and depth). That is, each layer takes as input a 3D volume of numbers and outputs a 3D volume. What one imagines as a 2D input image ($\boldsymbol{W} \times \boldsymbol{H}$) will be converted into 3D by introducing color depth as the third dimension ($\boldsymbol{W} \times \boldsymbol{H} \times \boldsymbol{d}$). The transformation does this with the help of two different types of layers.

The first is the convolutional (Conv) layer and we can think of it as a set of learnable filters. Suppose we have $\boldsymbol{K}$ such filters. Each filter is spatially small, with a range denoted by F, but extends to the depth of its input. For example, a typical filter might be $\mathbf{3} \times \mathbf{3} \times \mathbf{3}$ (F = 3 pixels wide and high, and 3 from the depth of the input 3-channel color image).

The second important type of layer in convnets is the pooling layer. It acts as a downsampling filter. They are used to reduce computational costs and also to some extent reduce overfitting. For example, a max pooling layer with spatial extent $\boldsymbol{F} = \mathbf{2}$ and stride $\boldsymbol{S} = \mathbf{2}$ reduces the input space by half from 4 × 4 to 2 × 2, keeping the depth the same.

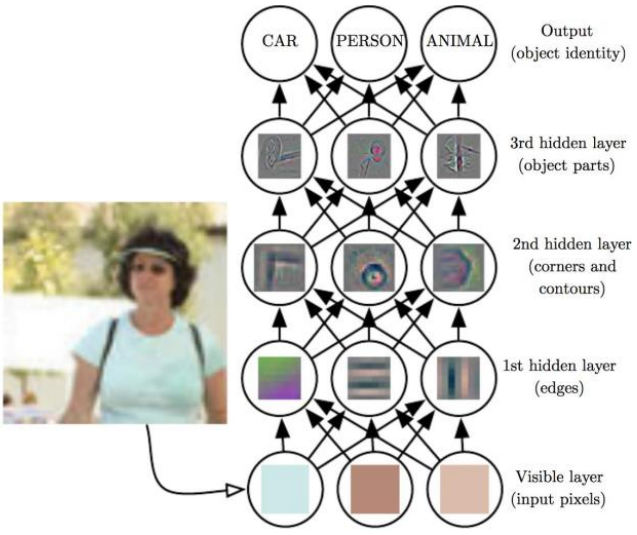

Fig. 5. Operation of CNN.

With raw pixel inputs, the first hidden layer is excited by simple features like edges, the next layer might be things like contours, and the next layer might be simple shapes and parts of objects. And the deeper they go, the more they capture the entire input field, not just a narrow region, but more importantly, the closer they get to a representation that makes it easy to classify.

It turns out that convolutional neural networks are all about learning representations. This is what makes them so powerful.

III. Style Conversion Problem

A. Problem Formulation

We have a content image $\mathbf{C}$ and a style image $\mathbf{S}$, and the core of this effect is to somehow extract the style (color and texture) from $\boldsymbol{S}$ and apply it to the content (structure) from $\mathbf{C}$ to arrive at the example output type shown below.

Fig. 6. Three-part image, showing the style image, $\mathbf{S}$, the content image, $\mathbf{C}$, and the transformed image, $\mathbf{X}$.

We can formulate this problem as an optimization problem:

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \left[\alpha L_{content}(\mathbf{C}, \mathbf{X}) + \beta L_{style}(\mathbf{S}, \mathbf{X}) \right] \quad (1)$$

That is, we are looking for an image $\mathbf{X}$ that is as little different as possible in content from the content image $\mathbf{C}$, and as little different as possible in style from the style image $\mathbf{B}$. The only drawback to this scheme is that we do not have a convenient way to evaluate the differences between the images in terms of content and style.

CNN models pre-trained for image classification have learned to encode the perceptual and semantic information we need to measure these semantically distinct terms. When learning object recognition, the network must be invariant to all image variations that are not necessary for object recognition.

B. Pre-trained Model VGG19

VGG is a complex neural network, born in 2015 and introduced at the conference ICLR 2015. The architecture of this model has many different variations: 11 layers, 13 layers, 16 layers, and 19 layers.

VGG19 uses a Conv-Conv-Conv-Conv sequence in the middle and end of the VGG architecture. This will make the computation longer but will still retain more features than using MaxPooling after each convolution. VGG19 consists of

19 layers: 13 Conv layers (02 layers conv-conv, 03 layers conv-conv-conv-conv) all have 3×3 kernels, after each convolutional layer is maxpooling downsize to 0.5, and 03 fully connected layers. Thus, with an input image (Input image) the size will be gradually reduced through convolutions (conv) but at the same time will increase the depth of the image. Thanks to that structure, VGG19 in general or VGG in particular is preferred for use in Object Detection (object detection).

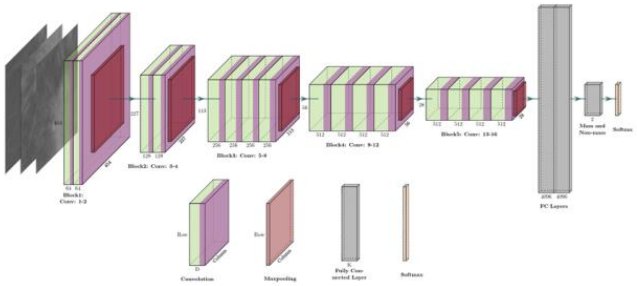

Fig. 7. Structure of VGG19.

IV. Problem Solution

A. Content Loss

Loss Function is a function that allows determining the level of difference of the result after each transformation process (Transform) compared to the target image value. Loss Function is a method of measuring the quality of the model after each transformation on the observed data set, binding the output and input parameters together. From there, it determines the operation of the system.

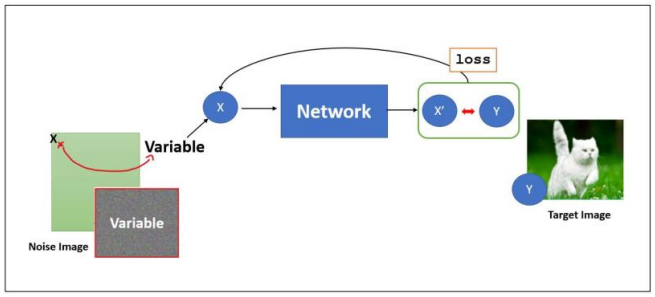

Fig. 8. Example of loss function.

The example shown in Figure 8, we have a Noise Image, in which the pixel values can be changed (acting as Variables), our goal is to design a Network so that each time ***X***' is updated, the more it moves towards the features of ***Y***, the Loss Function now plays the role of assessing the similarity of ***X***' and ***Y***. So that means we will design a function ***Distance***(***X***', ***Y***) for Loss Function, which has a function to measure the deviation between *X*' and Target Image, based on that idea, we have simple operations to measure the deviation as follows:

- Absolute Error: $|X' - Y|$ (2)
- Mean Square Error: $(X' - Y)^2$ (3)

In this problem, we choose the operation $(X' - Y)^2$ to design the Loss Function for Content Image, also known as Content Loss.

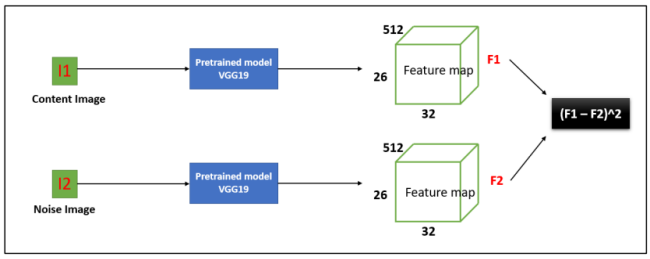

Fig. 9. Procedure to build the Content Loss.

In Fig. 9, I_1 is the Content Image and I_2 is the noise image. As mentioned above, designing a Content Loss function means that we build a certain Loss Function for the Content Image with the purpose of making the Noise Image after each update, it will approach the characteristics of the Content Image (I2 approaches I1), which means:

$$\boldsymbol{Content\ Loss \sim Distance(I2, I1) \rightarrow 0 \Leftrightarrow (F1 - F2)^2 \rightarrow 0} \quad (4)$$

Note: When designing the Content Loss function so that I2 approaches I1, it is still necessary to ensure that the spatial structure is preserved.

Here, the authors build the Content Loss function according to the Mean Squared Error Loss (MSE) method, also known as L_2 Loss, which is a Loss Function used for regression models, especially linear regression models.

Feature Map block: Generated after the process of determining the main features of the image in VGG19. Feature Map is a three-dimensional block structure (length of image ($\vec{p}$), width - number of channels (layers) of image ($\vec{l}$), height of image ($\vec{x}$)).

The Content Loss is then defined by

$$\mathfrak{L}_{content}(\vec{p}, \vec{x}, \vec{l}) = \frac{1}{n}\sum_{i,j}\left(F_{i,j}^{l} - P_{i,j}^{l}\right) \quad (5)$$

where *i* and *j* are the row and column indices of the considered image at layer *l*.

The authors build a model for the problem based on Pretrained Model VGG19, where we do not take the top layer of VGG19, but take the layers near the end to give better results (see Fig. 10).

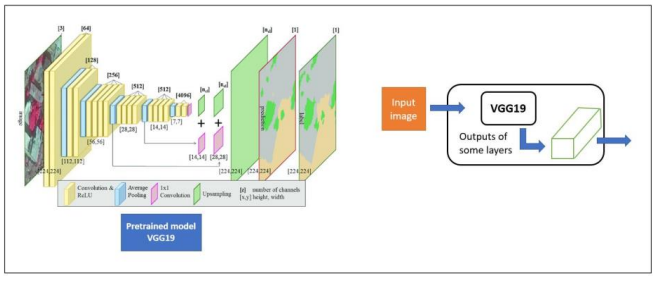

Fig. 10. Build a pre-trained model based on VGG19.

Therefore, according to the figure above, we will design a model that contains VGG19, with the input being the Input image, and the output being the last few layers of VGG19, the result of the Model will be the corresponding Feature Maps.

The Content Loss is built by using Python with the support of Keras.

B. *Style Loss*

Building a Style Loss function means we build a certain Loss Function for the input is a Style Image, executing a certain operation of the system [8].

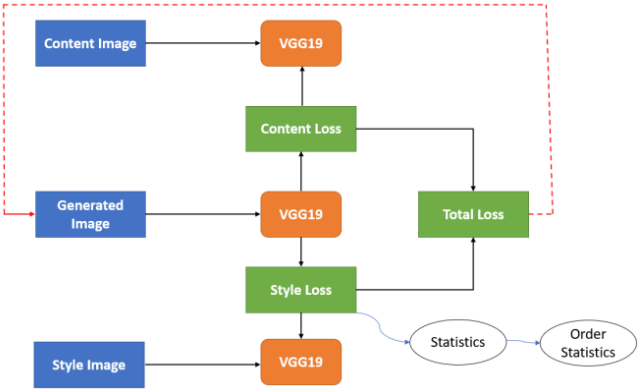

Fig. 11. Neural Style Transfer Overview Diagram.

After designing the Content Loss function, the remaining task is just to design the Style Loss function. The idea is completely similar to designing the Content Loss function. But the problem here is: we do not want to keep the spatial structure of the Style Image, our purpose is just to get the features of the Style Image, and mix them somewhere in the Content Image image. Therefore, in addition to design the VGG19 model as in the Content Loss function, to design the Style Loss, we need to find a mathematical operation that breaks the spatial structure of the Style Image. In the Order Statistic branch, the author chooses the Dot Product operation to do this job. Thus, just to design the Dot Product operation for the Feature Maps after going through VGG19, we can build the Style Loss function:

$$\textit{Dot product of } \vec{\mathbf{u}} \textit{ and } \vec{\mathbf{v}} = \sum_i u_i v_i \qquad (6)$$

Dot product (d) is also a form of convolution, considering the correlation between 02 objects. If we take each pair of vectors of the Feature Map to calculate their Dot Product, arrange them into 1 matrix, then that matrix is called Gram Matrix, we just need to take Gram Matrix and calculate Style Loss. To understand more clearly, let's go through the following example: Stlye Image after going through Pre-model VGG19, will generate 01 Feature map, for example block5_conv1 (26, 32, 512). As shown in Fig. 12a, block5_conv1 has 512 channels – each channel has 02 dimensions (26x32). Consider each channel as 01 vector ($n_H \times n_W$), then there are 512 corresponding vectors (because there are 512 channels). By taking each pair of vectors in these 512 vectors to calculate the dot product, the result will be arranged into Gram Matrix (512x512) in Fig. 12b.

Therefore, the formula of Gram Matrix is

$$\mathbf{G} = \mathbf{v} * \mathbf{v}^T \qquad (7)$$

So, we know how to calculate Gram Matrix, combined with the idea of building Content Loss function, we can easily build Style Loss function, instead of calculating Loss Function based on Feature maps generated from VGG19, we need to transfer those Feature maps into Gram Matrix (breaking the spatial structure), then calculate Loss Function as usual.

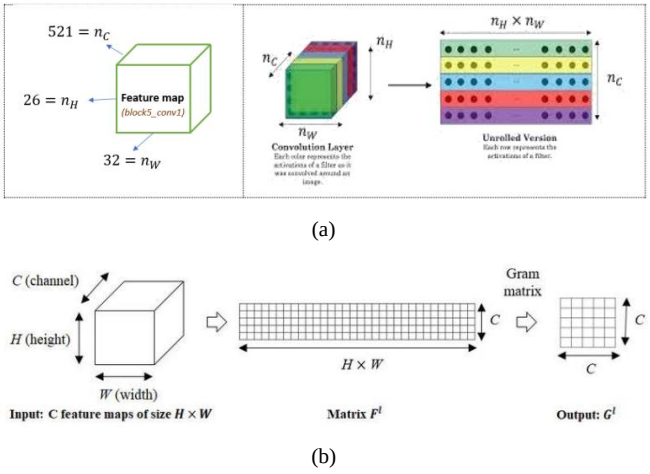

Fig. 12. Gram Matrix calculation.

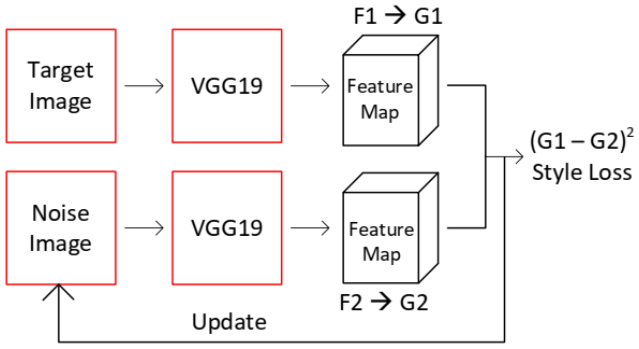

Fig. 13. Style Loss function design.

C. *Combining Style Loss and Content Loss*

So, in the previous sections, we have successfully built the Content Loss and Style Loss functions, the final task is to combine the two functions by adding them together, and perform the update, after many trainings, the Noise Image will gradually transform, to have the outstanding features and spatial structure of the Content Image, along with that they will be stylized according to the style of the Style Image.

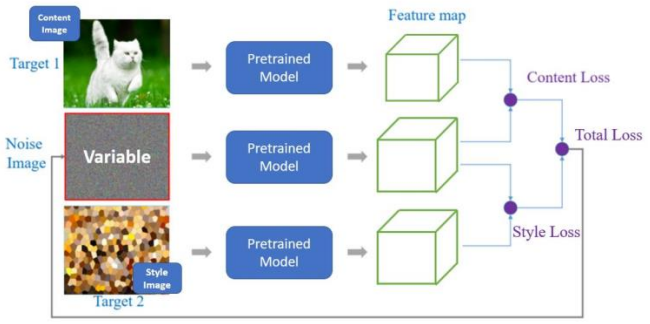

Fig. 14. Full flow-chart of NST algorithm.

V. RESULTS

A. *Content Loss*

After the training process, we get the results as shown in Fig. 15. Depending on the training time, the results will be different. We can easily see that, based on the Model and the designed Content Loss function, Noise Image will

continuously update and transform to have the characteristics of Content Image.

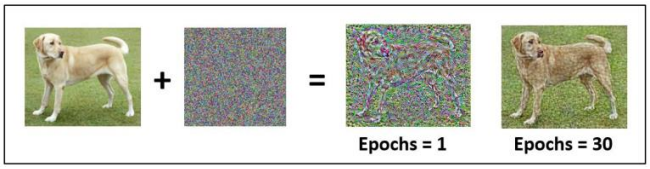

Fig. 15. Content Loss result using one layer of CNN.

To get more accurate results, we should combine using multiple layers together, which will give more accurate results. The results are shown as shown in Fig. 16. We see that it is quite similar to the original Content Image when combining using multiple layers.

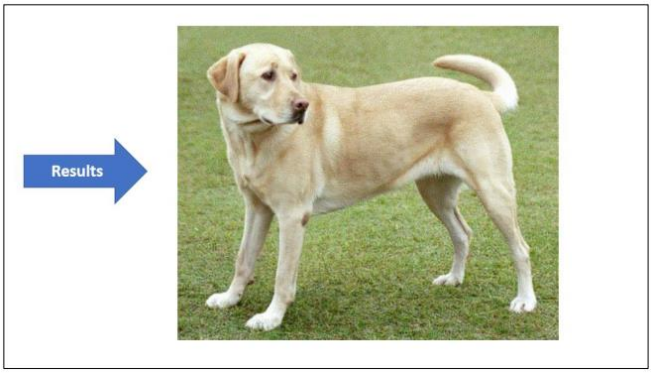

Fig. 16. Content Loss result using three layers of CNN.

B. *Style Loss*

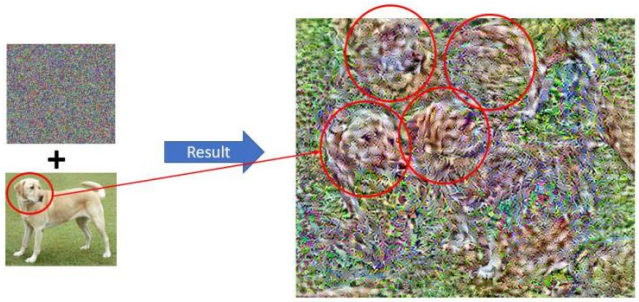

Fig. 17. Style Loss Function for Content Image.

As shown in Fig. 17, after running the Python program, we get relatively good results. The output has the prominent features of the input, along with the spatial structure has changed. Fig. 18 also shows the result of Style Loss for Style Image.

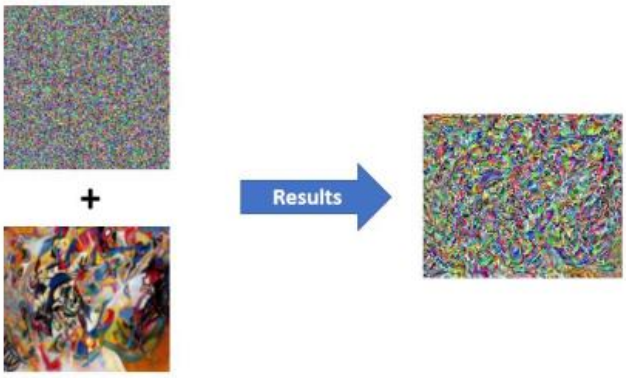

Fig. 18. Style Loss Function for Style Image.

C. *Style Conversion Results*

Fig. 19 illustrates the style-converted image. We can see that the artistic image has been created successfully. Fig. 20 shows the results for different input images.

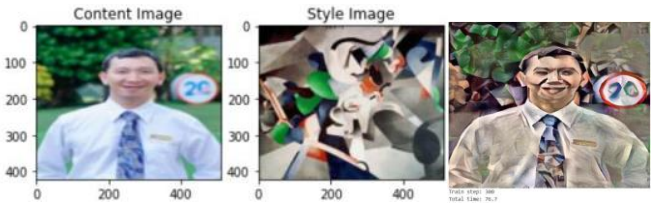

Fig. 19. Result afte training and testing.

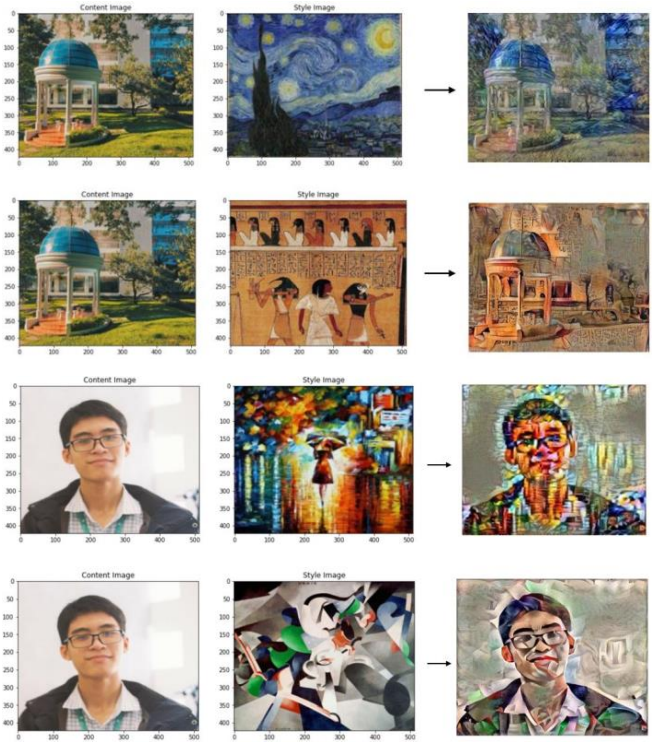

Fig. 20. Content Loss result using three layers of CNN.

VI. Conclusion and Future Work

Neural Style Transfer can help people enjoy the joy of creating and sharing artistic masterpieces, where users provide content images and artistic style image and from there the model will create artistic masterpieces as desired by the user.

To further optimize our model, we will need to adjust the weights based on a careful study of the distribution of content images with style images (these weights can be very different for each type of image) [9]. Choosing a style image (Style Image) is also a difficult and tricky task because not all styles are suitable for the original image. The general formula for this problem is

$$\mathfrak{L} = \alpha \mathfrak{L}_{content} + \beta \mathfrak{L}_{style} \tag{8}$$

Another direction is to change the operation for Gram Matrix. Instead of dot product, we can exploit other transformation like linear, polynomial, Gaussian, or Batch Normalization (BN) [10].

REFERENCES

[1] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu and M. Song, "Neural Style Transfer: A Review," in IEEE Transactions on Visualization and Computer Graphics, vol. 26, no. 11, pp. 3365-3385, 1 Nov. 2020, doi: 10.1109/TVCG.2019.2921336.

[2] N. Q. Tuyen, S. T. Nguyen, T. J. Choi and V. Q. Dinh, "Deep Correlation Multimodal Neural Style Transfer," in IEEE Access, vol. 9, pp. 141329-141338, 2021, doi: 10.1109/ACCESS.2021.3120104

[3] H. -H. Zhao, P. L. Rosin, Y. -K. Lai, M. -G. Lin and Q. -Y. Liu, "Image Neural Style Transfer With Global and Local Optimization Fusion," in IEEE Access, vol. 7, pp. 85573-85580, 2019, doi: 10.1109/ACCESS.2019.2922554.

[4] D. Marmanis, M. Datcu, T. Esch and U. Stilla, "Deep Learning Earth Observation Classification Using ImageNet Pretrained Networks," in IEEE Geoscience and Remote Sensing Letters, vol. 13, no. 1, pp. 105-109, Jan. 2016, doi: 10.1109/LGRS.2015.2499239.

[5] L. A. Gatys, A. S. Ecker and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 2414-2423, doi: 10.1109/CVPR.2016.265.

[6] G. Singh, K. Guleria and S. Sharma, "A Deep Learning-Based Pre-Trained VGG19 Model for Optical Character Recognition," 2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI), Coimbatore, India, 2024, pp. 801-805, doi: 10.1109/ICoICI62503.2024.10696044.

[7] Y. -F. Li, L. -Z. Guo and Z. -H. Zhou, "Towards Safe Weakly Supervised Learning," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 1, pp. 334-346, 1 Jan. 2021, doi: 10.1109/TPAMI.2019.2922396.

[8] H. Ye, W. Liu and Y. Liu, "Image Style Transfer Method Based on Improved Style Loss Function," 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Chongqing, China, 2020, pp. 410-413, doi: 10.1109/ITAIC49862.2020.9338927.

[9] T. Jeong and A. Mandal, "Flexible Selecting of Style to Content Ratio in Neural Style Transfer," 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 2018, pp. 264-269, doi: 10.1109/ICMLA.2018.00046.

[10] H. Ye, W. Liu and Y. Liu, "Image Style Transfer Method Based on Improved Style Loss Function," 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Chongqing, China, 2020, pp. 410-413, doi: 10.1109/ITAIC49862.2020.9338927.