Photo Style Transfer

Ngan Vo
University of Science
VNU-HCM
Vietnam
vhngan@apcs.vn

Son Nguyen
University of Science
VNU-HCM
Vietnam
nhson17@apcs.vn

Ngoc Mac Bao Nguyen
University of Science
VNU-HCM
Vietnam
nmbngoc@apcs.vn

Nhat Huynh
University of Science
VNU-HCM
Vietnam
hmqnhat@apcs.vn

***Abstract*—Style transfer is a popularly studied subfield in computer vision. Such technique gives anyone the power to produce beautiful artwork, inspired by their favorite paintings, textures and etc. This motivates our proposal to reimplement and re-evaluate the source code [1] written by Anish Athalye in TensorFlow, which follows the description in the research paper [2] by Gatys et. al. in CVPR 2015. The authors conduct experiments on our collected dataset with 100 images for input and another 100 ones for reference. We show that the more parameters we change, the more outputs transform.**

I. Introduction

For thousands of years in the past, converting an image from its own style to another favoutite style requires an excellent artist, lots of efforts and time. Since then, the art theories of styled artworks have absorbed attention of not only the artists but also plenty of computer science researchers. Neural Style Transfer based on Convolutional Neural Networks (CNNs) [6, 7, 11, 17] is remarkable from a technology perspective and is worth understanding. The main idea behind is to take two images, a content image and a style image. The Neural Style Transfer algorithm use these to create a third image that combines the content of the former with the style of the later. Thanks to that, a lot of studies and researches on how to automatically convert images to styled artworks are built. Among them, the advances in non-photorealistic rendering (NPR) [1], [2], [3] are born. However, most of these NPR stylisation algorithms aredesigned for particular artistic styles [3], [4] and cannotbe easily extended to other styles. In the community ofcomputer vision, style transfer is usually studied as a gener-alised problem of texture synthesis, which is to extract andtransfer the texture from the source to target [5], [6], [7], [8].Hertzmann et al. [9] further propose a framework namedimage analogies to perform a generalised style transfer bylearning the analogous transformation from the providedexample pairs of unstylised and stylised images. However,the common limitation of these methods is that they onlyuse low-level image features and often fail to capture imagestructures effectively.

Recently, inspired by the power of Convolutional Neural-Networks (CNNs), Gatys et al. [10] rst studied how to usea CNN to reproduce famous painting styles on naturalimages. Fig. 1 below shows an example: given a content image 1(a) and a style image 1(b), a stylized image 1(c) is generated.

Fig. 1. Example image style transfer

This result is built based on CNNs: while the high-level CNN features encode the semantic content, e.g. the objects and global structure of the input image, whereas the low-level features encode the local details, e.g. colors, basic shapes and texture. Neural Style Transfer specifies a total loss for a new image, which consists of both the content loss, i.e. its high-level feature difference with the content image, and the style loss, i.e. its low- level feature difference with the style image. An image minimizing the total loss will combine the global semantic content of the content image and the local texture of the style image.

The seminal work of Gatys et al. has attracted wideattention from both academia and industry. In academia, lots of follow-up studies were conducted to either improveor extend this NST algorithm. The related researches of NSThave also led to many successful industrial applications(e.g., Prisma [11], Ostagram [12], Deep Forger [13]). However, there is no comprehensive survey summarising anddiscussing recent advances as well as challenges within thisnew eld of Neural Style Transfer.

In this paper, we aim to provide an overview of Gatys's Neural Style Transfer (NST). First, we guide through steps of NST. Second, we re-implement the method on our own dataset. Third, evaluate by altering iterations, some parameters, and image size.

(Cha cite xong cc reference !!!)

II. Method

Overview. Since no dataset is published, our approach begins with creating it. The data containing content images and reference ones go through Neural Style Transfer model to produce combined artworks. Then those artworks are taken into consideration to re-evaluate by altering iterations, some parameters, and image size. Figure 2 illustrates the overall framework.

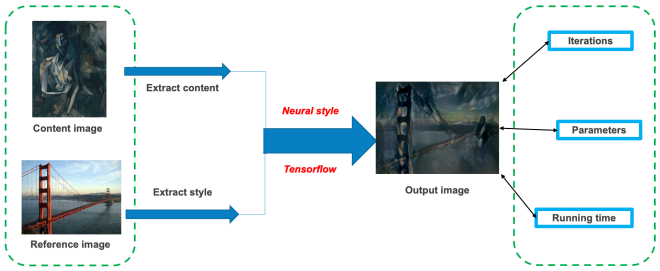

Fig. 2. Photo Style Transfer Framework

Background. In order to fully explain Gatys et al. paper, we have to detail the three parts of the workflow: the content extractor, the style extractor and the merger.

Let's begin with the content extractor. We need a way to separate the semantic content of an image. Fortunately there is already a special kind of deep neural network suited for this job, called the convolutional neural network (CNN). This neural net has an architecture that is inspired by some of the mechanisms present in our visual system and its main use is in computer vision problems.

It is known that the hidden layers of a convolutional neural network extract high level features of an image (the deeper the layer, the more high level the attributes will be that the layer identifies). Knowing this, the researchers took a pre-trained VGG 19 neural net, made some slight tweaks in the weights to adapt to their problem and then used the output of one of the hidden layers as a content extractor. In figure 3, the image of a winter wolf goes through that process of content extractor.

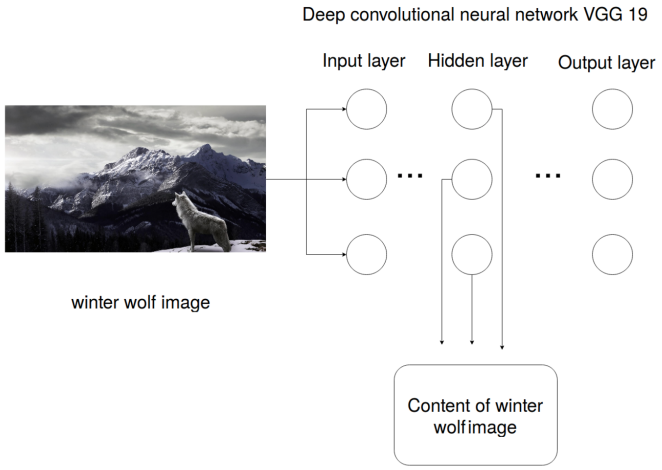

Fig. 3. Content Extractor

The style extractor use the same idea of the content extractor (i.e. use the output of a hidden layer), but it adds one more step. It uses a correlation estimator based on the Gram matrix of the filters of a given hidden layer. Which is just a complicated sentence that means: it destroys semantics of the image, but preserves its basics components, making a good texture extractor. In figure 4, the famous Scream painting ...

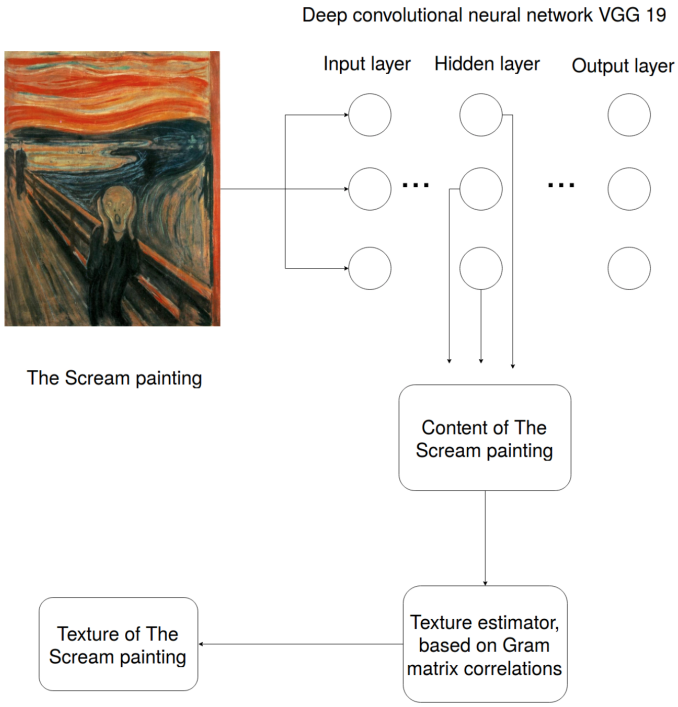

Fig. 4. Style Extractor

As the final step, we need a way to blend the content of one image with the style of another. This is done with the help of an optimization problem. An optimization problem is a mathematical problem where we define a cost function, which we want to minimize. Generally, the cost function decreases in every iteration of the optimization, assuming it has not arrived at a minimum. Therefore, we can expect that each iteration of the optimization gets our cost function closer to our goal.

$$\mathcal{L}_{\text{total}} = \sum_{\ell=1}^{L} \alpha_\ell \mathcal{L}_c^\ell + \Gamma \sum_{\ell=1}^{L} \beta_\ell \mathcal{L}_s^\ell$$

$$\text{with: } \mathcal{L}_c^\ell = \frac{1}{2N_\ell D_\ell} \sum_{ij} (F_\ell[O] - F_\ell[I])_{ij}^2$$

$$\mathcal{L}_s^\ell = \frac{1}{2N_\ell^2} \sum_{ij} (G_\ell[O] - G_\ell[S])_{ij}^2$$

Fujun Luan et. al. gives short notes on this function: I is the total number of convolutional layers and ℓ indicates the ℓ-th convolutional layer of the deep convolutional neural network. In each layer, there are N_ℓ filters each with a vectorized feature map of size D_ℓ.$F_\ell[\cdot] \in \mathbb{R}^{N_\ell \times D_\ell}$ is the feature matrix with (i, j) indicating its index and the Gram matrix $G_\ell[\cdot] = F_\ell[\cdot]F_\ell[\cdot]^T \in \mathbb{R}^{N_\ell \times N_\ell}$ is defined as the inner product between the vectorized feature maps. α_ℓ and β_ℓ are the weights to configure layer preferences and is a weight that balances the tradeoff between the content (Eq. 1b) and the style (Eq. 1c).

Now, how can we use optimization to merge arbitrary content with arbitrary style? Gatys et al. approach was to create a cost function which penalizes the synthesized image if its content was not equal to the desired content and its style was not equal to the desired style.

Re-evaluating experiments. The authors design some experiments to evaluate the method by altering altering iterations, some parameters, and image size.

We first investigate the quality and running time of the output images by doing the iteration changes. We use only content image of a lamb and style image of yellow chics. Both of them have size of 800x600 pixels. There has an example of them and their output in figure 5. We do 20 experimens starting with 500 iterators and extend it extra 500 ones for each time. 10000-iteration experiment is the last one to do. The result of these experiments can be found in Table ..., and example retrievals in Figure... We now highlight some of the takeaways.

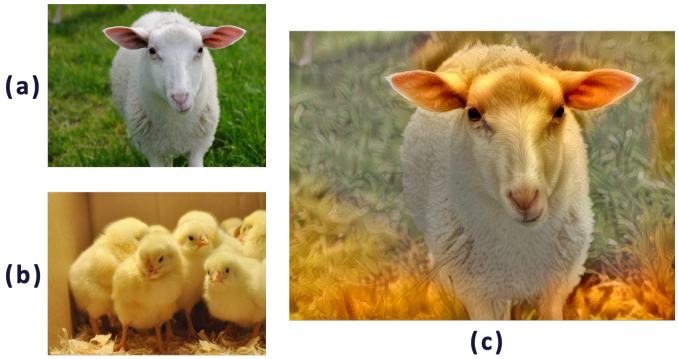

Fig. 5. ...

III. Experimental results

(This section is not finished yet!!!)

A. Collecting data

Since no dataset is published so the authors created it by ourselves. The basic problem is to combine 2 pictures together to produce a new image, the data is very diverse and easy to find. Here we collect 10 different topics for input images with 10 photos each. Some of the topics are portrait, animals etc. Basically, the input and reference dataset share a common so 100 input images is used as reference ones as well.

B. Re-evaluate

1) Iterations altering: We notice of a special parameter named *iteration*. The more iterations, the more time it takes and the more beautiful the output are. We test many cases with unlike iterations. Running it for 500-2000 iterations seems to produce nice results. In terms of running time, it takes 8 mins 32 secs for 1000 iterations and 31 min for 3000 iterations with the same input and reference images.

2) Parameter altering: –content-weight-blend [1] specifies the coefficient of content transfer layers. Default value - 1.0, style transfer tries to preserve finer grain content details. The value should be in range [0.0; 1.0].

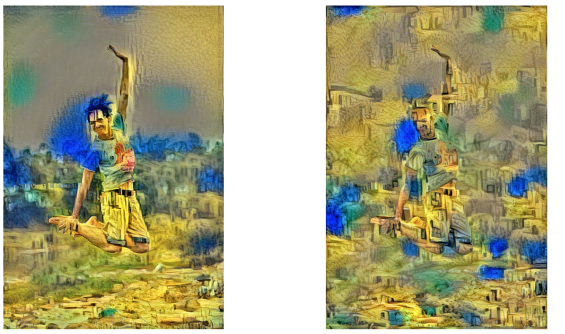

Fig. 6. –content-weight-blend

–style-layer-weight-exp [1] command line argument could be used to tweak how ”abstract” the style transfer should be. Lower values mean that style transfer of a finer features will be favored over style transfer of a more coarse features, and vice versa. Default value is 1.0 - all layers treated equally.

–pooling [1] allows to select which pooling layers to use (specify either max or avg). Original VGG topology uses max pooling, but the style transfer paper suggests replacing it with average pooling.

–preserve-colors [1]

IV. Conclusion

(This section is not finished yet!!!)

Fig. 7. –style-layer-weight-exp

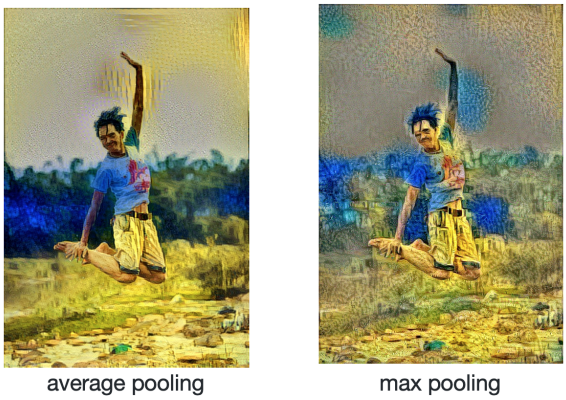

Fig. 8. –pooling

REFERENCES

[1] Anish Athalye. Neural style. https://github.com/anishathalye/neural-style, 2015. commit xxxxxxx.

[2] Santiago A. Cadena, Marissa A. Weis, Leon A. Gatys, Matthias Bethge, and Alexander S. Ecker. Diverse feature visualizations reveal invariances in early layers of deep neural networks. *CoRR*, abs/1807.10589, 2018.

Fig. 9. –preserve-colors