Neural Style Transfer: Using Art to create Images in the same Style

Cornelius Wolff, Juri Moriße, Annika Richter

Abstract: This paper is an implementation of an artistic style transfer approach based on "A Neural Algorithm of Artistic Style" by Gatys, Ecker and Bethge (2018). We start by giving a short explanation of how the transfer of artistic style functions, followed up by giving a brief overview over different approaches towards the challenge of artistic style transfer. We continue with a more detailed explanation of the pre-trained vgg network and loss functions that were used in the original paper as well as by us. Subsequently, we describe our own implementation of a similar network and present the results we obtained with it. In the following, we present and compare results we obtained by using pre-trained networks and evaluate how different parameters influence the result. Additionally, we investigate the inverse application of making art more realistic and use our results to point out the limitations of our approach. In the end, we give a summary of our findings and outline over possible future research.

Keywords – Neural Style Transfer, Convolutional Neural Network

1. Introduction

Recreating Art can be Art itself. Stylistic ways to create artistic images can be a unique way that can resemble an artist long throughout their lives. Creating images in a specific artistic style by using a convolutional neural network has been successfully done by Leon A.Gatys , Alexander S. Ecker and Matthias Berthge in the paper "A Neural Algorithm of Artistic Style" published in 2015. Using a pre-trained VGG Network their algorithms can create a new Image based on a content-defining image and a style-defining image. In this paper, we recreated their original Algorithm using the pre-trained model and training our own VGG Model.

2. Different Approaches

2.1 Approaches without using Neural Networks

To give a broader perspective on the topic of computational style transfer, we will start by briefly presenting two approaches to this task that do not rely on artificial neural networks to solve this problem, the stroke-based rendering approach and the region-based abstractions approach.

Stroke-Based Rendering

One of the first digital approaches to transfer the style of a work of art to an original image was introduced by Haeberli in 1990 and is now known as Stroke Based Rendering. This approach does not use neural networks to transfer the style. Instead, the user simply paints on a digital image canvas that has the same size as the canvas of the image. However, the user is not able to influence the color of the stroke used, only the size. The orientation of the stroke is also part of

the algorithm and is calculated based on an edge detection system at the corresponding image location. The color of the brush is determined by an algorithm that tries to recreate the corresponding color tone and image content used in the original photo. This results in a highly abstracted version of the photo, as the user can paint in a similar way to classic paintings and thus determine the style, while the actual content is taken from the original photo. In summary, this is not a complete automation of the style transfer, but only a partial automation since the user's input is still required to achieve the target image. Jing, Yang, Feng, Ye, Yu, Song (2018)

Region-Based Abstractions

Region-based abstractions are another approach to the problem of transferring a certain style of an artwork to a different picture. With this approach, the image is divided into different regions depending on color, contours or content. The size of the different regions can vary significantly. For example, the sky can be grouped into one region, while individual stones or leaves can also represent individual regions. Once the picture has been divided, it can be transformed into a mosaic based on the colors, shapes, etc. in each region. A major problem with the approach is that when the picture is automatically divided into regions, errors often occur, resulting in making the style transfer very error-prone. Although there are some techniques, such as smoothing, to limit this problem, there is not yet a generalized working solution to fix this. However, if it is possible to find a well-functioning solution in the future, this could be of great support for creating animations, for example. Jing et al. (2018)

2.2 Approaches using Neural Networks

According to Jing et al. (2018) the main challenge in transferring an artistic style with neural networks is to represent the style of an image. While not completely, a large part of the information about the style of an image is contained in the information about its texture. This allows to circumvent the problem of computing style representations by using texture representations for which are easier to obtain based on previous work. Therefore, Yang et al. distinguishes different neural network approaches based on how they represent the texture of an image. They describe two different concepts. The approach used in Gatys et al. (2016) and by us is part of what Ling et al. call ‘Parametric Neural Methods with Summary Statistics. The core idea here is that the texture of an image can be represented by using summary statistics over all the image’s pixel values. In our case, the summary statistic in use is the Gram matrix which describes the correlations between filter responses in the VGG networks. The alternative take on texture representation Ling et al. call ‘Non-Parametric Neural Methods with Markov Random Fields’. Here the texture is not represented based on all pixel values but instead an assumption is made that a certain pixel value is dependent on the pixels around it. Thus, the texture representation is based on an analysis of several smaller neighborhoods of pixels instead of the global analysis over all pixels used in the parametric approach. Ling et al. point out that parametric approaches are prone to losing spatial information because of their use of global statistics. This leads to poor performance when modeling regular or symmetrical textures as well as for realistic styles. While non-parametric approaches counter those weaknesses, they struggle if the content and style inputs are not similar in shape and perspective.

3.0 The VGG Network

In the paper “A Neural Algorithm of Artistic Style”, a VGG Network consisting of 16 convolutional and 5 pooling layers was used. This VGG Network is a convolutional Network that performs remarkably well on image classification tasks. Its architecture consists of five blocks consisting of a pooling layer followed by three convolutional layers.

In the original VGG Network, a MaxPooling Layer was used, but in the referenced paper, they were able to obtain better results with AveragePooling Layers.

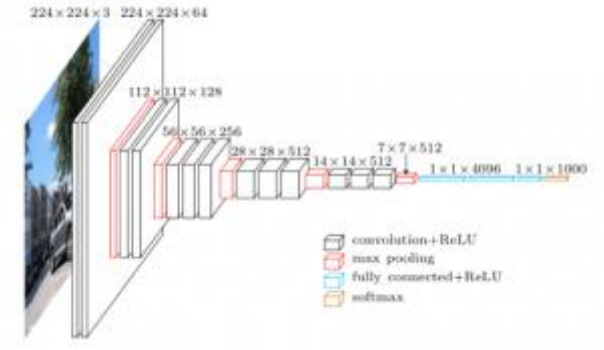

Figure 1

There are pre-trained models for both the VGG16 and the VGG19 network available that include numerous advantages. These models are trained to classify the images of the Dataset ILSVRC 2012, more frequently known as ImageNet. ImageNet is a dataset consisting of over 15 million labeled images that are ordered according to the WordNet hierarchy. A high amount of time and computational power is necessary to train the VGG Model for a lot of epochs on this huge dataset. Using the pre-trained model allows us to train with less computational power and a smaller amount of time on our Loss functions without having any disadvantages in performance.

4.0 The Loss functions

To find an image that uses the content of the content input and the style of the style input, gradient descend is performed on a simple noise picture. The Network tries to find the picture that matches best the content features of the content input and the style features of the style input. That is accomplished by minimizing two error losses. The loss corresponding to the content is defined by a squared Euclidian distance between the two feature representations:

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l}\right)^2$$

($\vec{p}$ and $\vec{x}$ being the original images, P_{ij}^{l} and F_{ij}^{l} their features representations in layer l)

The loss corresponding to the feature correlations between the two images is defined by the mean-square distance between the gram matrices of the two images. Therefore, the Style Loss minimizes the differences between the Matrix of the style input and the generated image, ensuring that the generated image matches the style of the style input. In order to do so, the contribution of the style representation of the original and the generated image in each layer are summed up and weigthed to get the final Loss function.

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

($\vec{a}$ being the original image, $\vec{x}$ the image that is generated, w_l a weighting factor and E_l the contribution of the style representations to the loss in layer l)

The total loss that is minimized to create a stylized image of the content image is the weighted sum of these two loss functions.

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

($\vec{p}$ being the original content image,$\vec{a}$ the original style image, $\vec{x}$ the image that is generated and $\alpha \; and \; \beta \; weighting \; factors \; for \; style \; and \; content$)

5. Own Implementation of VGG13

5.1 Training

In the beginning, we started by implementing the VGG network used in the paper itself. However, throughout the developmental period we decided to implement only the smaller VGG13 network. This has two main advantages in our case: first, it reduces the computational cost by significantly reducing the number of parameters that can be trained in the model. Second, due to the limited computational power of Google Colab, we decided to train the model using only the Cifar100 dataset and not the ImageNet dataset. Since the images have a much lower resolution and there are only 100 different classes with 6 images each, the simpler VGG13 network is also sufficient for successful classification. Consistent with the Style Transfer paper, an AveragePooling-Layer was used after each CNN block.

Our assumptions were confirmed during the training over 30 epochs. Thus, the model achieved an accuracy of about 99.95 percent. For better learning behavior, the learning rate was reduced by a factor of 10 from 0.001 to 0.0001 after 15 epochs. Adam was used as the optimizer and categorical cross-entropy as the loss function. The batch size was 128.

5.2 Style Transfer

However subsequently, we encountered the problem that we had to adapt the model in such a way that the correct style and content layers are output during the style transfer and that the previously trained weights can be applied at the same time. For this reason, the model is only defined in the constructor of the VGG13 network, but not compiled. Depending on the usage, one can either call the compile() function to train the model. Alternatively, one can call the

import_weights_for_style_transfer() function, which will then import the specified weights and select the correct output layers. Afterward, you can use the model for the style transfer with the help of the loss functions described before. We used 1000 iterations as parameters, set the Content_Weight to 5000 and the Style_Weight to 0.001. As an optimizer, we used Adam with a learning rate of 5 with beta_1 being 0.99 and epsilon 0.1. The results are shown below:

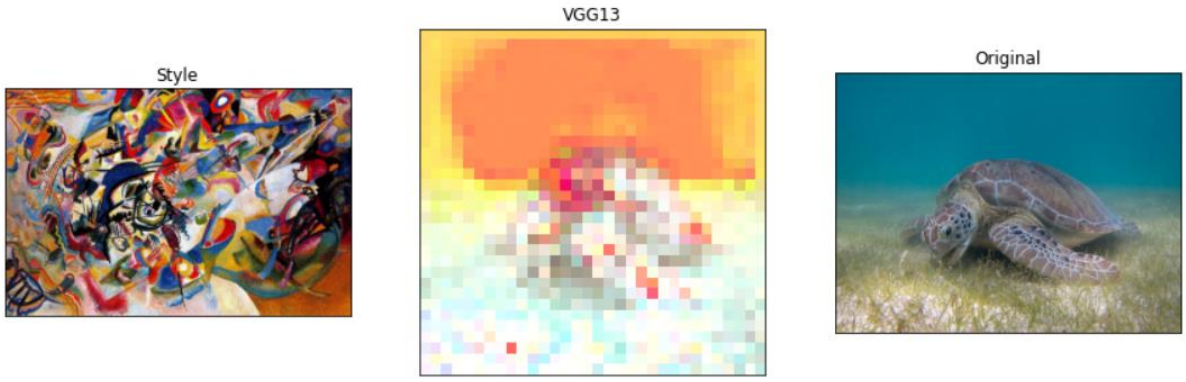

Figure 2

Due to the Ciphar100 dataset used for training, the result of the style transfer is also in a rather low resolution of 32x32 pixels. Nevertheless, you can clearly see from the calculated image that the style transfer was successful. In order to display the image in a higher resolution, the model would first have to be trained with the ImageNet dataset. Fortunately, appropriately pre-trained models are provided by TensorFlow for the VGG16 and VGG19 networks.

5. Using pretrained VGG16 and VGG19

Using the pre-trained models we were able to make the Style Transfer in a higher resolution which allows us to be able to compare the results of using the VGG16 and the larger VGG19 model. The parameters we used were a content weight of 5 and a style weight of 0.001. We gave the network 1000 iterations to perform the style transfer and again used Adam as the optimizer with the same parameters as with the VGG13 model. The results show that the style transfer process profits significantly from using the larger and deeper pretrained VGG19 model:

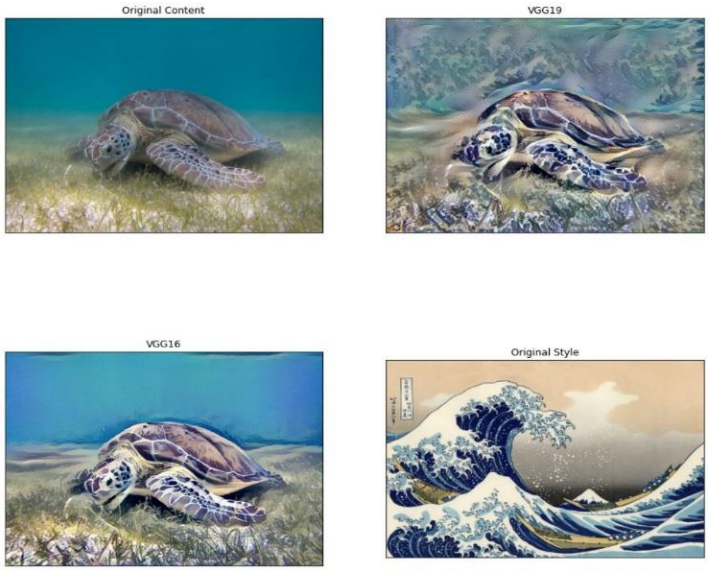

Figure 3

6.1 Attempt to make the artistic styles more realistic

After we obtained the previously outlined results, we thought about other possible applications. After some consideration, we wanted to see if it is possible to switch content and style images and if by doing so one can obtain a more realistic version of the artistic image. For this purpose, we used a picture of a grazing sea turtle as a style image and Japanese woodcut 'The Great Wave off Kanagawa' as content image (Figure 4 a) and b)). However, the resulting images (Figure 4 d)) were not more realistic but rather the opposite as they showed a greenish glow presumably coming from the color of the sea turtle image.

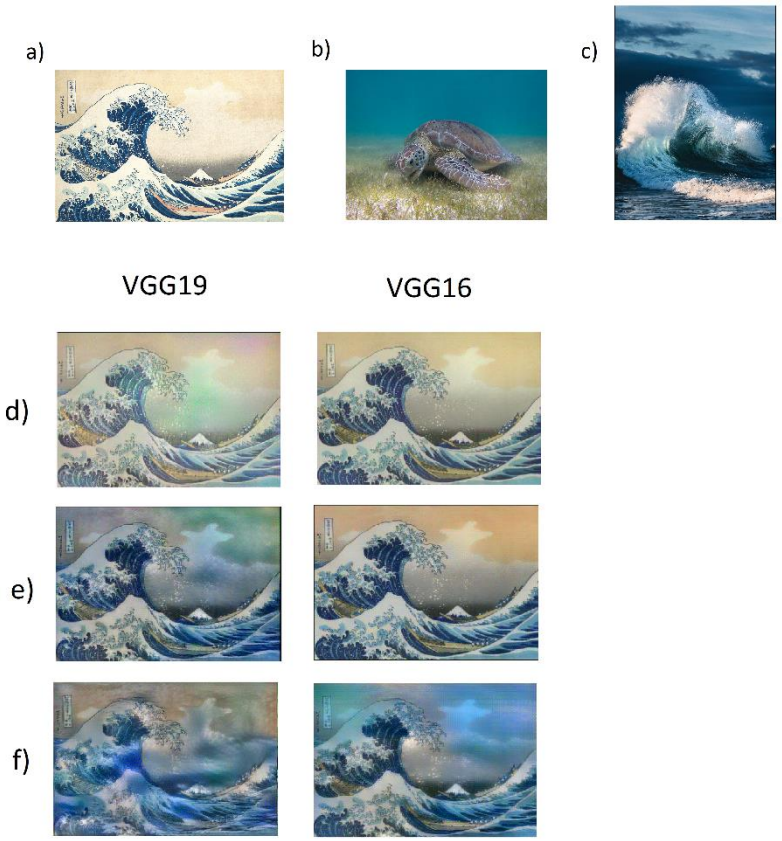

Figure 4

To obtain a result with more realistic colors, we changed the style image and used a photograph of an ocean wave (Figure 4 c)) instead. This led to the first promising results (Figure 4 e)) as it shifted the colors to more realistic tones of blue and blurred the sharp contours that the content image had due to it being a woodcut. But still, we were not quite satisfied with the result as the images still seemed very graphic and not so realistic. We were able to still improve the result by increasing the style weight. While this led to the disappearance of the boats (when using VGG19), it preserved the waves and showed more realistic colors, a more realistic texture and blurred the sharp contours (Figure 4 f)).

We also tried this with a self-portrait of Van Gogh as content and a portrait photograph as style image (Figure 5 a) and b)). While the results showed more realistic skin tones and a reduction in Van Gogh's characteristic swirls (Figure 5 c)), it did not work as well as with The Great Wave of Kanagawa. There are several reasons why this inverse application of our approach shows only limited success in both cases. One problem is that it relies on the image classification network. When we use an artistic image as the content image, the underlying network needs to compute a content representation and for that it needs to identify objects in the content image. Because the networks we used are trained on photos, it is understandable

that they struggle with classifying objects that are shown in an artistic style. This is most likely the reason for the disappearance of the boats seen in Figure 4 f). A second problem was already outlined previously in the overview of approaches using neural networks. According to Jing et al. (2018) approaches using global statistics like the gram matrix to represent style struggle with regular texture and symmetry in the style picture. This could explain why the result for the Great Wave off Kanagawa was more realistic. The picture of the wave shows very irregular texture and no symmetry while the texture of the portrait photograph used for the Van Gogh portrait is regular and contains symmetrical structures. Additionally, Ling et al. state that the use of global statistics is generally not suitable for realistic styles. It would be interesting to see if an alternative non-parametric approach using Markov random fields would perform better in this inverse application.

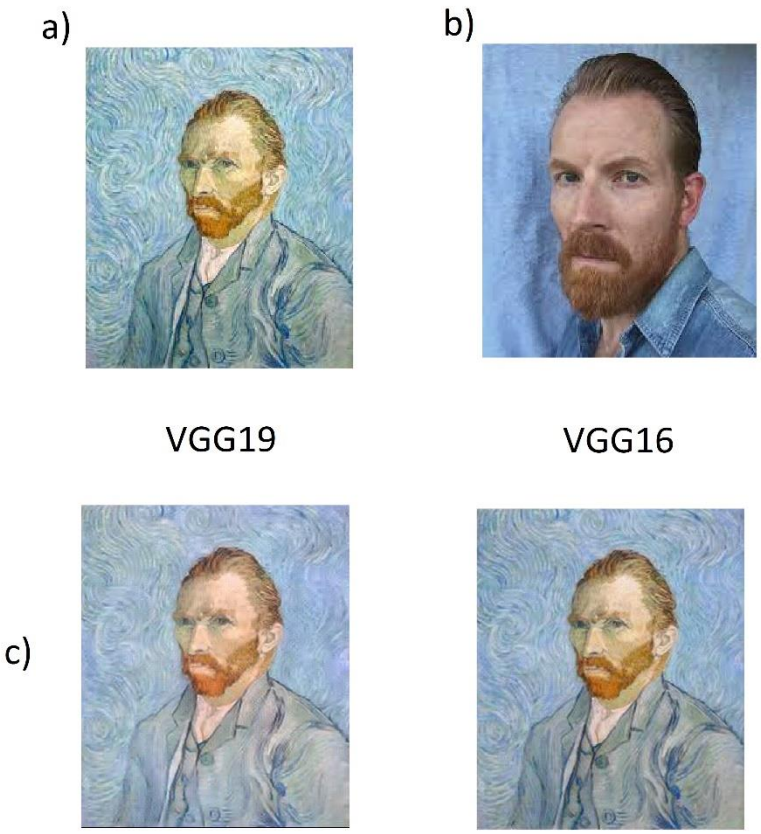

Figure 5

7. Conclusion

In summary, we have succeeded in reimplementing the paper described at the beginning and also succeeded on generating our own results in the form of a new style and content image and the inverse use of style and content images. Furthermore, the implementation of our own VGG13 network serves as a possible foundation for conducting further experiments with the network later on and observing the resulting consequences for the style transfer. However, much larger computing resources will be needed for these experiments, as training on the 150GB ImageNet is likely to be required for this purpose.

Sources

Gatys, Leon A.; Ecker, Alexander S.; Bethge, Matthias: A Neural Algorithm of Artistic Style; https://arxiv.org/pdf/1508.06576.pdf

Hokusai (1830): The Great Wave off Kanagawa, last checked 04.04.2021; https://en.wikipedia.org/wiki/The_Great_Wave_off_Kanagawa#/media/File:Tsunami_by_hokusai_19th_century.jpg

Jason Fenmore: Ocean Waves Photography, last checked 04.04.2021; https://www.pinterest.com.au/pin/624944885772338137/

Jing, Yongcheng; Yang, Yezhou; Feng, Zunlei; Ye, Jingwen; Yu, Yizhou; Song, Mingli: Neural Style Transfer: A Review; https://arxiv.org/pdf/1705.04058.pdf

Lindgren (2013): Green Sea Turtle Grazing Seagrass, last checked 04.04.2021; https://commons.wikimedia.org/wiki/File:Green_Sea_Turtle_grazing_seagrass.jpg

Petter Samuelson (2016): Portrait Photograph of Van Gogh Lookalike, zuletzt geprüft am 04.04.2021; https://images.app.goo.gl/vGmcuyZ25XBddu9y7

Simonyan, Karen; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition; https://arxiv.org/pdf/1409.1556.pdf

Van Gogh (1889): Selbstportrait, last checked 04.04.2021; https://de.wikipedia.org/wiki/Datei:Self-Portrait_(Van_Gogh_September_1889).jpg

Vassily Kandinsky (1913): Komposition 7; https://de.m.wikipedia.org/wiki/Datei:Vassily_Kandinsky,_1913_-_Composition_7.jpg