Artistic style transfer between images using Convolutional Neural Networks

Sergio López, Mónica Villanueva, Diego Yus
sergiol@kth.se, monicavi@kth.se, diegoyl@kth.se

Deep Learning in Data Science DD2424

Abstract. This project uses transfer learning to perform style transfer between a photograph and a painting with a pretrained discriminative Convolutional Neural Network. The responses to both the painting and the photograph in the different layers are used selectively to define style and content losses. These are afterwards combined and weighted to define a final total loss that is to be optimized using back propagation by modifying an input image instead of the weights of the network, in order to finally obtained the desired new image.

Keywords: CNN, style transfer, transfer learning

1 Introduction

1.1 Motivation

Training a Deep Neural Network (DNN) from scratch is a very time-consuming and not always rewarding task. However, it is possible to use already trained networks to solve different problems in a wide variety of domains by means of transfer learning. Since the authors are working in parallel in a speech domain DNN project it was a good idea to focus this one on computer vision.

Though useful, tasks like object recognition are very mainstream and less appealing in a project like this one. Choosing style transfer as the topic for this project provides the excuse to work in a less common deep learning problem using an already trained network. This gives the opportunity to focus on the underlying concepts and analysis results instead of wasting time in never-ending training executions.

1.2 Problem description

The main objective of this problem is to transfer the style of any paint to any selected photograph, so that the result resembles a painting by the treated author, but its content is actually the given photograph.

The style transfer problem is based on the assumption that the content and the style from a picture can be isolated. Once done that, it is possible to merge this two representations regardless of their source. This way, DNN are able to create artistic images.

This project describes an attempt to replicate the most famous paper on style transfer by Gatys et al. (2015) [1].

In this paper, the authors describe an algorithm that performs style transfer using an already trained discriminative Convolutional Neural Network, (dCNN), in this case, VGG-19 [2]. The breakthrough of their work is proving, indeed, that content and style are separable and can be obtained using a dCNN.

Convolutional Neural Networks are DNN that target different levels of abstraction of the input in each layer by using filters. The deeper the layer in the network, the higher-level the representation will be, focusing on the content, instead that on low level pixel structures. The representation of the responses in the higher layers will provide the content for the generated image.

On the other hand, the style can be described as the spatial correlation between the different filter responses in one layer. This style can be smoothed by computing a weighted mean over different layers. In a similar way, the style reconstruction represents the texture, but not the elements populating the image. Again, the higher the layers involved, the larger the size and complexity of the texture features. This is appreciable in Figure 4.

Mathematical Formulation The graph have different loss functions for style and content that are combined later to compute the total loss.

First, knowing that the content is simply the responses of the filters in a certain layer (l), the content loss can be computed as the square error between the content image (p and its filter responses P) and the generated one (x and its filter responses F).

$$loss_{content}(p, x, l) = \frac{1}{2} \sum_{i.j} (F_{ij}^l - P_{ij}^l)^2 \tag{1}$$

The style loss is a bit less direct. Firstly, the Gram Matrix correlating all the responses in a layer (l) must be computed for each filter (i, j) and position (k) as

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \tag{2}$$

The loss of the layer is

$$E_l = \frac{1}{4N_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \tag{3}$$

with N being the multiplication of the three dimensions of the filter responses, G the Gram Matrix of the generated image and A the Gram Matrix of the style image.

Each layer that takes part in the style is weighted to compute the style loss between the paint (a) and the generated image (x)

$$loss_{style}(a, x) = \sum_l w_l E_l \tag{4}$$

Then, both losses from Equations 1 and 4 are combined in the global loss function, with two parameters that control the balance between the importance of the style and content terms in the global loss equation.

$$loss_{global} = \alpha \cdot loss_{content} + \beta \cdot loss_{style} \tag{5}$$

where α and β are two parameters to control the balance between the style and content.

The minimization of Equation 5 is used to find an optimal combined image using the two representations, that minimizes the global loss.

2 Background

Image synthesis is wide field in which style transfer is contained. This area has become a hot topic in recent years since the publication of Gatys et al. [1] in 2015. Being one of the first papers to present differential results, is a reference in the area and it has been widely cited in all the posterior works, that build on their idea. Since then, more methods have been published trying to improve certain points on this work, that has already been explained in Section 1.2.

The fist paper that provided an alternative technique was that of Li et al. [3]. They included the Markov Random Filters (MRF) regularization to increase the plausibility of the features being transferred, avoiding the hallucinatory artifacts present in the previous work [1]. The proposed algorithm learns the distribution over smaller patches of the image assuming that the most important dependencies are local. However there is a limitation on the adaptation of these patches. It also needs a high-level constraint in order to learn the global arrangement.

In the discussion they highlight the restrictions and improvements of using MRF in combination with dCNN. MRF works better when there is are good pattern matches between the content and the style, but also creates artifacts when there is no pattern to match in the content image. There are some examples of pictures where the output of both methods ([1][3]) are compared. However, it is difficult to measure whether the outperformance of one network or the other is real or it is influenced by confirmation bias.

Almost one year after the publication of their well-known work, Gatys et al. published an improvement on their previous idea, *Preserving Color in Neural Artistic Style Transfer* [4]. The work is centered on two techniques that boost the visual perception of similitude: keeping the color and luminance of the content image.

For the color transference they compare two methods, both of them based on transferring the color distribution of the content to the style image using linear transformations on the mean and variance of the RGB values. After that, the basic style transfer algorithm is performed over the content and the modified style image.

To transfer luminance, the luminance channel is extracted from the content and style image and blended using the style transfer algorithm. Using the YIQ format, it is possible to combine the generated luminance channel. Nevertheless, if the luminance is too different in the content and style images, it is useful to apply a linear transformation to the style luminance, similar to the one implemented for color.

Despite all this improvements, the output image still displays features that reminds of a painting, even when using a photorealistic input as the style image. This issue is tackle by Luan et al. [5] proposing two solutions for the main challenges.

In order to preserve the structure they use a photorealism regularization term that penalizes distortions, profiting from the fact that the style image is already photorealistic. For that they perform transforms that are locally affine in color.

On the other hand, regarding the problem of semantic accuracy and transfer faithfulness they implemented an optional guidance system based on semantic segmentation so that style is only transferred between segmentation masks that match in label, e.g. buildings, water, etc.

It has been said lately that Generative Adversarial Networks (GANs) will be the next important development in deep learning. This triggers the interest of researchers to apply them to any problem at hand, image synthesis included. In the paper published by Liu et al. [6] GANs are used to auto-paint black-and-white sketches. This is not exactly style transfer, but they share a hidden and common goal: generate art using DNNs. This method allows the automatic selection of artistic coloring but also allows the user to select their selection.

GANs are composed by a generator that learns the distribution and a discriminator that predicts if a sample comes from the training set or from the generator. This particular net is a conditional GAN, in which the generator learns from input images instead of only noise. GANs are trained as a Mini-Max game in which the generator tries to minimize the loss function and the discriminator tries to maximize it.

In their implementation they used VGG-16, but in order to insert some constraints, they developed a 'U-net' architecture that connects layers in the encoder with layers in the decoder to keep the edges and the quality of the picture. They also implemented a control model to fit the tastes of different users using color block that fill specific region. Subsequently, they used a evaluation metric base on the subjective aesthetic preferences of the volunteers.

Finally, it is worth mentioning that using a pretrained CNN is not an original technique used only in this task. Other problems such as detection and location [7] have benefited from the use of networks like VGG.

3 Approach

3.1 VGG-16

In the aim to replicate the original paper, the same network architecture was used. VGG-16 is a CNN architecture firstly proposed by K. Simonyan and A. Zisserman [8] from the Visual Geometry Group from Oxford, that was trained on the ImageNet dataset and performed extremely well on the ILSVR (ImageNet) competition in 2014 [9].

To reduce the number of parameters, all convolutional layers make use of small 3x3 filters with the stride set to 1. Besides, ReLU is used as the activation function and all the pooling layers use 2x2 pooling with stride 2. According to the original authors, average pooling produces more pleasant images than max pooling, so average pooling is used instead. A representation of the whole net is depicted in Figure 1.

It is important to notice that, since we are not interested in classification but only in the responses of the convolutional layers, the last 3 fully connected layers are not included in our model. This removes the constraint for fixed input size (224x224) and allows for arbitrary input sizes so it is possible to generate higher resolution images.

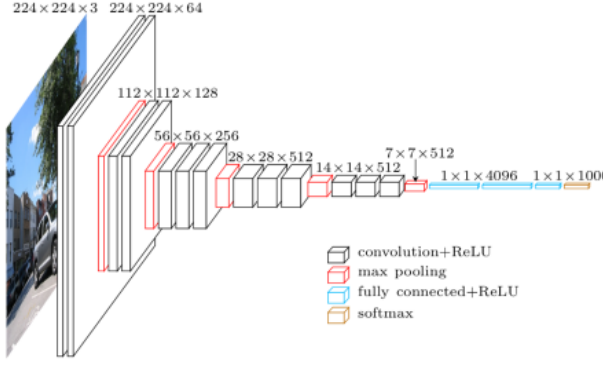


Fig. 1. Architecture of VGG16

Since the original model was released by the authors in Caffe, and this project is implemented in TensorFlow, the converted weights are obtained from Davi Frossard [10].

It is important to notice that the original paper used the VGG-19 network, that includes one more convolutional layer before each of the last 3 pooling layers. After trying both architectures, the differences were insignificant and the more satisfying results were obtained using the VGG-16 network, so it was decided to use it for the whole project.

3.2 Computational graph

TensorFlow has a visualization tool that allows the automatic generation of the computational graph. However, after inspecting this auto-generated diagram, it was decided to make a more friendly though simplified version of the graph (Figure 2).

The red nodes are parameters of the algorithm, black nodes represent variables and the blue node is the output image that is being learned in order to minimize the loss function. The arrows symbolize the operations to go from one variable(s) to its child/children.

Note that the names of all variables follow the description in Section 1.2.

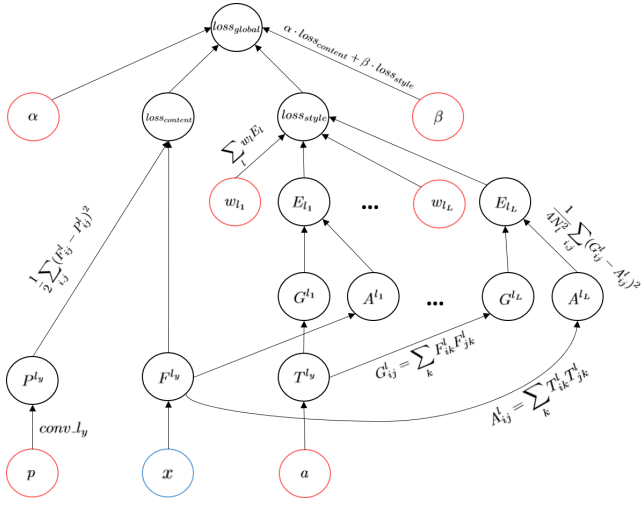


Fig. 2. Simplified computational graph for the style transfer algorithm

3.3 Experimental parameter setup

The actual implementation for this project will be available in a Github repository[1] under the accounts of the authors. The data set used are images as close as possible to those employed in the original paper [1], since they do not point to any place to get the same dataset.

In this specific project, as opposed to other CNNs experiments, the training variables are not the weights of the net (already pre-trained) but the generated image.

The experiments are designed to meet the following goals:

[1] `https://github.com/MonicaVillanueva/CNN_Style_Transfer`

1. Replicate the results from the original paper
2. Understand the effect of different parameters
   - Layers: The selection of the layers for content and style set a different starting point for the generation. As explained before, for content the aim is to get the general arrangement while style is supposed to contribute with the texture. To understand completely what is generated, the reconstructions from each layer in content and style are displayed in Figure 4.
   - α/β ratio: Represents the relation between content and style. As it is shown in Section 4.3, playing with this parameter alters greatly the appearance of the output images. Several ratios are tested, differing by one order of magnitude, to show its effect (Figure 5) and then chose the value that produced the most pleasant image to perform further experiments.
   - Annealing strategy: The Adam Optimizer implemented in TensorFlow is used for the SGD. After performing a search over the learning rate space, it became apparent that the value of this parameter is not that critical as long as its value is in a reasonable range.

4 Results and conclusions

To reach the convergence point where the total loss is minimized, the initial point is a noisy random image that is further trained. However, if the initial image is not a noisy one, but the content image (photograph) instead, the convergence process is accelerated, because the starting point is an image closer to the final desired output. Therefore, for the generation of the artistic images (Section 4.4), the training is started from the content image. In the reconstruction experiments (Section 4.3) however, the images are reconstructed from a noisy image.

4.1 VGG-16 vs VGG-19

It has been said in Section 3.1 that after using both networks it was decided to keep VGG-16 for the experiments opposite to the implementation on the paper being replicated. The design decision was driven by the more aesthetic appealing of the results obtained with this architecture.

To support this claim, two images generated with both architectures are included so that the reader can also judge compelled by his/her own artistic taste (Figure 3).

4.2 Learning rate annealing strategy

Independently of the type of optimizer used, it is possible to choose between several exponential decays for the learning rate such as exponential decay, or inverse time decay. Furthermore, the learning rate accepts arguments to choose its initial value, decaying rate, the frequency of the decay and what they call

(a) VGG16 (b) VGG19

Fig. 3. Comparison of generated images for both network architectures

staircase, that allows the decrease to be done at discrete intervals.

Our final set up included Adam optimizer with exponential decay. The initial learning rate was usually set to 8, except for some specific reconstructions from the noisy image, and decayed with a factor of 0.94 every 100 steps. Finally, the staircase flag is set to True in order to decay the learning rate at discrete intervals. This helps avoiding getting stuck in local minima.

These values are variable depending on the visual experience sought and the style image, mostly (see more details in Section 4.4).

Regarding the evolution of the content and style losses, as we normally give a higher importance to the style loss (β is higher than α), the network tends to decrease the style loss at the cost of the content loss, in order to minimize the global loss.

In our experiments, mainly due to computational limitations, all images were obtained starting from the original photograph, instead of doing it from a noisy image, as the authors of [1].

For this reason, the value of the content loss starts being zero, and it afterwards grows as the style content diminishes. Their values, even after obtaining the desired output image, do not come close to zero, but rather tend to converge in $1 \times 10^7 - 1 \times 10^9$ for images of size 448×448.

4.3 α/β ratio

In order to understand how the emphasis on each component of the image works, it is useful to reconstruct the content and style along different layers.

In Figure 4, the content and style reconstructions from a noisy image for the different layers are observable. In the style reconstructions, it is possible to notice that for higher layers, the model reconstructs patterns of larger size, because it tends to a higher abstraction level due to the increase in size of the receptive field. In the content reconstructions, we are able to reconstruct the original image with very small differences for the first three layers, because the

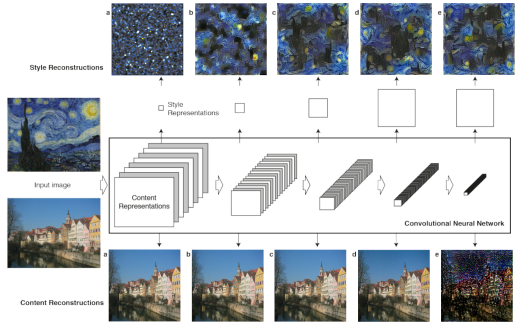


Fig. 4. Reconstruction of content and style for the different layers.
In the content reconstructions: "a": *conv*1_1, "b": *conv*2_1, "c": *conv*3_1, "d": *conv*4_1 and "e": *conv*5_1.
In the style reconstructions: "a": *conv*1_1, "b": *conv*1_1 and *conv*2_1, "c": *conv*1_1, *conv*2_1 and *conv*3_1, "d": *conv*1_1, *conv*2_1, *conv*3_1 and *conv*4_1, "e": *conv*1_1, *conv*2_1, *conv*3_1, *conv*4_1 and *conv*5_1.
Note: The template of the image belongs to the original paper but the images are generated with our network configuration

model works on small details and pixel level. However, for the last two layers, where the model focuses on higher abstraction features, it is more difficult to reconstruct the original image.

The results displayed in Figure 5 illustrate an insight on how the style representations at each layer differ from each other, but also the power of control that the ratio α/β comprise in the recombination.

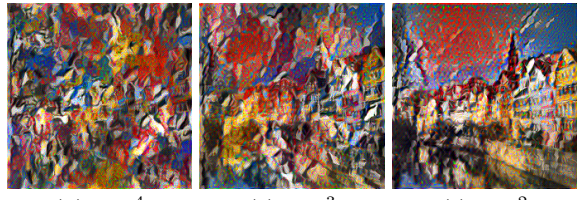
The top row of the grid of images represents the images obtained using the style representations of only the first layer. The following rows are the style representations from the cumulative inclusion of layers into the optimization.

As it can be observed, the obtained images resemble a lot the photograph if the style is chosen from the lowest layer, specially using high values of ratio α/β. This trend seems to be inverted as we include more and more layers into the style optimization definition, making the image more abstract and needing of quite low α/β values in order to make it possible to appreciate the original photograph lines.

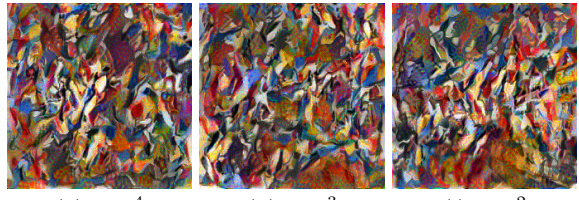
4.4 Individual experiments

Using a certain original photograph, shown in 6(a), the style of different painters is applied to it while maintaining the content component. The new paintings were obtain using a α/β ratio of 10^{-3}, and slightly different hyper parameters, such as the learning rate η depending on the characteristics of each of the individual styles.

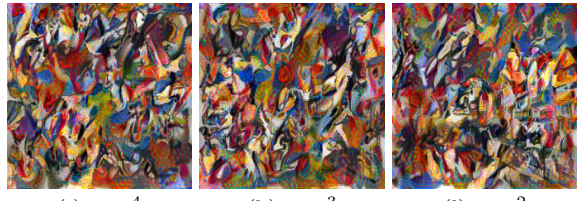
(a) 10^{-4} (b) 10^{-3} (c) 10^{-2}



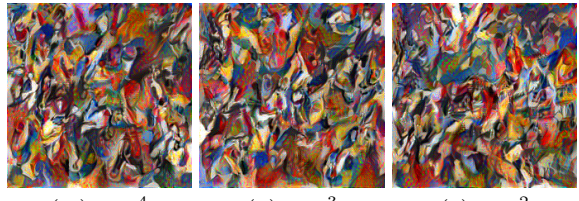
(d) 10^{-4} (e) 10^{-3} (f) 10^{-2}



(g) 10^{-4} (h) 10^{-3} (i) 10^{-2}



(j) 10^{-4} (k) 10^{-3} (l) 10^{-2}



(m) 10^{-4} (n) 10^{-3} (o) 10^{-2}

Fig. 5. The first row uses only the layer *conv*1_1 for the style, and the following rows add more layers until finally using *conv*1_1, *conv*2_1, *conv*3_1, *conv*4_1 and *conv*5_1 at the lowest row of the image.
The number in each column indicates the ratio between α and β.

(a) Original (b) Turner (c) Van Gogh

(d) Munch (e) Picasso (f) Kandisky

Fig. 6. Different painter styles applied to the same original image.

The images obtained are comparable to those presented in the original paper [1]. However, it is difficult to obtain the same visual experiences without knowing their parameter setting completely and without using the same architecture.

4.5 Video

As an original idea, it is possible to apply this technique to videos. At the end of the day, video is only a succession of images fast enough to fool the brain into thinking it is a moving picture.

There are two possible techniques that can be applied so that the final output is a video showing style transfer.

The first method is creating a video from the intermediate representations of the generated image. From a didactic point of view, it is interesting to see how the learning process is happening and how the loss is being minimized in terms of the image being generated.

The procedure is simple: saving the intermediate representations of the output image when transfer is taking place. Then, bringing all this images together fast enough to see an appreciable change.

An example of this evolution can be found as a GIF file in the project repository [2].

The other alternative is applying the technique directly to a video. That way it is possible to convert a video content instead of an image.

However, the process is the same: the algorithm needs an image as input, so the video must be split into frames. The style is actually transferred into each one of the frames and when the process is over, the video is reconstructed from its modified parts.

Due to lack of resources, it has been impossible to test this theory in a successful way.

4.6 Conclusion

The goal of this work is to proof that content and style can be isolated and recombine using the algorithm proposed by Gatys et al. (2015) [1].

The results obtained using said method are comparable to those present in the reference paper and all the parameters that take part in the generated image are analyzed so that each output is understood under the influence of the experimental setting.

Our final contribution to the idea is the application of the method to streams of images in order to generate videos with a certain style.

[2] `https://github.com/MonicaVillanueva/CNN_Style_Transfer/blob/master/style_transfer/gen_images/kth_b%3D1000/Video.gif`

References

1. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576 (2015)
2. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
3. Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2479–2486
4. Gatys, L.A., Bethge, M., Hertzmann, A., Shechtman, E.: Preserving color in neural artistic style transfer. arXiv preprint arXiv:1606.05897 (2016)
5. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. arXiv preprint arXiv:1703.07511 (2017)
6. Liu, Y., Qin, Z., Luo, Z., Wang, H.: Auto-painter: Cartoon image generation from sketch by using conditional generative adversarial networks. arXiv preprint arXiv:1705.01908 (2017)
7. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 779–788
8. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
9. None: Vgg competition
10. Frossard, D.: Vgg in tensorflow