



# 097200 - Deep Learning Course Final Project Style Transfer

Lior Danino<sup>1</sup> and Daniel Goman<sup>2</sup>

<sup>1</sup> Technion Israel Institute of Technology, Haifa , ISR  
liordanino@campus.technion.ac.il, 

<sup>2</sup> Technion Israel Institute of Technology, Haifa , ISR  
danielgoman@campus.technion.ac.il, 

## 1 Introduction

### 1.1 Convolutional Neural Networks

Convolutional Neural Networks(CNN) is one of the leading Deep Neural Networks architectures used today for image processing task. Today it is rare to see any visual computation done without CNN. The CNN consist of layers of small computational units that process visual information hierarchically in a feed-forward manner. Each layer of units can be understood as a collection of image filters, each of which puts emphasis on a certain feature from the input image. Thus, the output of a given layer consists of feature maps, meaning differently filtered versions of the input image.

### 1.2 Style Transfer

Style transfer is a process of modifying the style of an image while preserving its content. Neural style transfer is an optimization technique used to take two images – a content image and a style reference image – and generate a new image using convolutional neural nets. To do so, we first need to extract content and style from the given images, and combine them in the output image. Common uses for neural style transfer are creation of artificial artwork from photographs, for transferring the appearance of famous paintings to user supplied photographs.

## 2 Deep image representations

The key finding of the paper is that it is possible to separate the style representation and content representation in a Convolutional Neural Net. Along the processing hierarchy of the network, the input image is transformed into representations that are increasingly sensitive to the actual content of the image, but become relatively invariant to its precise appearance. Thus, deeper layers of the network capture objects

and their relative position in the input image, while the lower layers are more focused on individual pixel values of the original image. We therefore refer to the feature responses in higher layers of the network as the content representation, and the feature responses in lower layers of the network as the style representation.

## 2.1 Content representation

Generally, each layer in the network defines a non-linear filter bank. A given input content image  $\vec{x}$  is encoded in each layer of the Convolutional Neural Network by the filter responses to that image.

The responses in a layer  $l$  can be stored in a matrix  $F^l \in \mathbb{R}^{N_l \times M_l}$  where  $F_{ij}^l$  is the activation of the  $i^{th}$  filter at position  $j$  in layer  $l$ ,  $N_l$  is the number of distinct filters (feature maps) and  $M_l$  is the size of the feature map (height×width). We will denote  $\vec{p}$  as the input (content) image and  $\vec{x}$  as the image that is generated, and  $P^l$  and  $F^l$  their respective feature representation in layer  $l$ . We can now define the **Content Loss** as the squared-error between the two feature representations:

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2 \quad (1)$$

The derivative of this loss with respect to the activations in layer  $l$ :

$$\frac{\partial L_{content}}{\partial F_{ij}^l} = \begin{cases} (F^l - P^l)_{ij} & , \text{if } F_{ij}^l > 0 \\ 0 & , \text{if } F_{ij}^l < 0 \end{cases} \quad (2)$$

The Content Loss ensures that the activations of the higher layers are similar between the content image and the generated image, meaning that the content from the original input image is captured in the generated image.

## 2.2 Style representation

In order to represent the style of an image, on top of the filter responses in each layer of the network, we compute the correlations between the different filter responses. These feature correlations are given by the Gram matrix  $G^l \in \mathbb{R}^{N_l \times N_l}$  where  $G_{ij}^l$  is the inner product between the vectorized feature maps  $i$  and  $j$  in layer  $l$ :

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \quad (3)$$

By including the feature correlations of multiple layers, we obtain a representation of the input image, which captures its texture information but not the global arrangement.

We will denote  $\vec{a}$  as the original image and  $\vec{x}$  as the image that is generated, and  $A^l$  and  $G^l$  their respective style representation in layer  $l$ . The contribution of layer  $l$  to the total loss is:

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

and the total **Style Loss** is:

$$L_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^L \omega_l E_l \quad (5)$$

where  $\omega_l$  are weighting factors of the contribution of each layer to the total loss. The derivative of  $E_l$  with respect to the activations in layer  $l$ :

$$\frac{\partial E_l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} \left( (F^l)^T (G^l - A^l)_{ji} \right) & , if F_{ij}^l > 0 \\ 0 & , if F_{ij}^l < 0 \end{cases} \quad (6)$$

The Style Loss ensures that the correlation of activations in all the layers are similar between the style image and the generated image. from the original input image is captured in the generated image.

### 2.3 Style transfer

To transfer the style of an artwork  $\vec{a}$  onto a content image  $\vec{p}$  we synthesize a new image  $\vec{x}$  (the image that is generated) that combines those two components together. The **Loss** function we minimize is:

$$L_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha L_{content}(\vec{p}, \vec{x}) + \beta L_{style}(\vec{a}, \vec{x}) \quad (7)$$

where  $\alpha$  and  $\beta$  are user-defined weighting factors for content and style reconstruction, respectively. To extract image information on comparable scales, we always resized the style image to the same size as the content image before computing its feature representations.

## 3 dataset

Since we used a pretrained VGG19 network, we didn't need a large varied dataset. Therefore, we decided not to use an existing dataset, but to use some images we found on google. We used three images for the content and three images for the style, in order to show the results of our model.

For the content images, we chose images with well noticeable objects:



Content images

For the style images, we chose artistic or view images, without specific objects:



Style images

## 4 Experimenting and evaluation of the results

### 4.1 Evaluation of the results

As of right now, it seems that there is no evaluation metric for the performances of the style transfer method, not one that is popular and widely used anyway. With that said, we decided that we'll manually evaluate the results according to the visual appearance of the generated images

It is important to note that extracting the style of an image is not a well-defined problem. This is problematic as style may be intuitive to figure for a specific picture, but there's no generalization of what defines the style of an image, nor a rule-of-thumb to identify it.

Thus, deciding whether the style of one image was correctly extracted and correctly applied to the content image isn't a trivial problem, and even visual appearance evaluation will probably be inaccurate.

### 4.2 Hyperparameter tuning

In order to optimize the results of the procedure, some hyperparameter needed to be tuned. We tried various values and methods for each hyperparameter and chose those who seemed to improve the results, as said, based on personal judgement of visual appearance of the results. Among those hyper parameters we had the following:

- Content weight and Style weight
- Amount of epochs
- Optimizer's learning rate
- Type of optimizer
- Type of model for the feature extraction
- Which layers will be used for content optimization
- Which layers will be used for style optimization

### 4.3 Content and style matching trade-off

Naturally, when combining the content of one image with the style of another image, it's quite impossible to retrieve a result that simultaneously perfectly matches both constraints. Even though a perfect synthesis is hard to achieve, we must note that the loss function is a linear combination of the loss functions for content and for style. Thus, we can quite easily regulate the emphasis both on content and style. A strong emphasis on content means that the generated image will be quite similar to the original image with not as recognizable appearances of the style. A strong emphasis on style means that the generated image will have an appearance of an artwork, effectively displaying the style of the style image, but the connection between the content image and the resultant images will be slim to none. Here we'll display some results, based on the various  $\alpha/\beta$  ratios we tested:



**Content image**



**Style image**

$e^1$



$e^{-2}$



$e^{-3}$



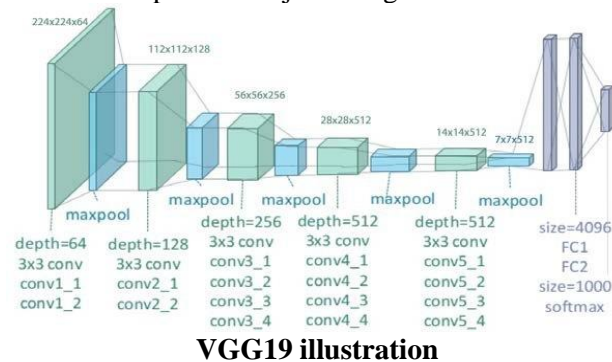
$e^{-4}$





#### 4.4 Preprocessing model selection

We chose to use a pretrained VGG19 model as our preprocessing model which was trained to perform object recognition and localization.



We tested several VGG models, such as VGG11, VGG13, VGG16 and finally VGG19. Although seemingly the outputs of these models were nearly visually identical, we settled on using VGG19, as in theory, VGG19 has a lot of layers, so it learns in different scales and so captures more details about the input/content. Other models down sample very quickly, and so they lose a lot of vital information. Interesting to point out, even though that the various VGG models vastly differ in the number of parameters they have, the run time of the models didn't seem to differ as much. Here are some visual results of our test:



**Content image**



**Style image**

*VGG11*

*VGG13*



VGG16



VGG19



#### 4.5 Content and Style loss layers selection

For content, in the original paper the second part of the forth convolutional layer was used, that being conv4\_2. We tested various layers as the layers for the content loss function: conv1\_1, conv2\_2, conv3\_2, conv4\_2, conv5\_2.

There wasn't a well noticeable difference in the appearance of the images. And yet, it is possible to spot some slight differences in the texture of the images. Usually, the differences appear mainly on the sides, as the main object in the content image (which usually can be found in the middle of the image) is being preversed. As mentioned before, in theory, higher layers should capture the content of the image, while the lower layers percept shallower features like colors and basic shapes:



**Content image**



**Style image**



*conv1\_1*



*conv2\_2*



*conv3\_2*



*conv4\_2*



*conv5\_2*



For style, in the original paper they used a weighted combination of [conv1\_1, conv2\_1, conv3\_1, conv4\_1, conv5\_1] altogether. The theory behind that is that since our intention is to capture the style of the image, the lower layer should do the job, as they focus on shallow features and don't capture specific objects. We'd also like to include some parts from the higher layer, since features we're interested in for style purposes might also be found in the higher layer. Then again, we're more interested in the lower layers, and thus we'll give them higher weights.



## 4.6 Final results

Here are some of the final results we have achieved on the images we chose to work with:

**Content image**

**Style image**

**Generated image**

1.



2.



3.

