CS 464 - Progress Report - Group 7
Neural Style Transfer of Paintings to Portraits and Selfies

Team Members: Ömer Ünlüsoy (21702136), Ata Berk Çakır (21703127), Doğa Tansel (21802917), Mustafa Hakan Kara (21703317), Ahmet Cemal Alıcıoğlu (21801700)

1. Introduction

Selfies have become a very common way of taking personal pictures in the recent decade. The commonality is most prominent in social media where people like to use many different aesthetic effects with the selfies they take. The same is true for portrait photography as well. Although portraits are used in much more formal settings like LinkedIn, they are also widely used. Selfies are more widely used among young people compared to adults and older people. In the end, both of these photo styles represent a very personal angle for the people in the picture, and so many people like to edit these photos using different techniques.

Style transfer is a method of applying the features of one input image (photo, painting, etc.) which is called Content Image onto another input image, Style Image, to produce an output image, Target Image, with the content of one and the style of the other. Style Transfer is a well-known and common practice in Machine Learning and the simplest versions can be used to teach the fundamentals of it. Moreover, this production of a new image from two different images is considered as an art since it allows the user to adjust the blend ratio and the artistic users can produce beautiful results.

In our project, we will be using Convolutional Neural Networks for the style transfer implementation. We will be transferring styles of famous paintings onto photographs in the form of portraits and selfies, especially focusing on selection of certain art movements like Fauvism, Impressionism, Expressionism, etc.

So far, we have been using the pre-trained VGG - 19 model from PyTorch to extract vital features from different layers of the images. We have been passing content and style images to this model to extract features from the different layers as previously mentioned. We are also smoothening the optimization procedure by calculating the gram matrices of the style features. Style losses are also part of our calculations, which are passed to the optimizer we use, Adam Optimizer, and reduced by it. The optimizer trains the final product on the target image, which means every pair of style-image and content starts the training procedure from scratch.

Afterwards, we will be reworking our procedure, as style transfer creates a large runtime issue as the optimizer runs on the images themselves rather than on the model. We have been working with two images so far, since this optimizer allows us to not use datasets yet. We will be moving onto our selected datasets in the near future. We will also be adding the implementation for selecting different art styles for style transfer. We will be trying and testing different models such as VGG19 and VGG16 along with AlexNet.

2. Background Research

In portraits and selfies, it can be difficult to capture and apply style transfer on features of two input images to create an acceptable output image. Some Convolutional Neural Network style transfer algorithms can find dense correspondence on faces by transferring local statistics of one image onto another one [1]. Dense correspondence can help match properties like local contrast or overall lighting direction but is not very efficient when it comes to reducing artifacts at a high level [1]. Dense correspondence techniques cannot establish the same correspondence or achieve appealing blending on other elements of an image, but there are more detailed techniques that can do this better [2].

Some algorithms use convolutional neural networks to create different mappings of different categories of features of images, such as maps of semantic, position and appearance maps [2] [3]. When convolutional neural networks are trained for specific tasks like object recognition with well-labeled data, there are very promising results. The neural networks can learn how to extract high-level image features and generalize them in such a way that texture recognition and artistic style classification is also done very well [3].

The convolutional neural network algorithm for style transfer tries to optimize a total loss function that is defined as a weighted sum of the style loss and the content loss [4]. The ratio of these weights is a hyperparameter called alpha ratio and can be played with in order to improve the target image.

Content loss and the style loss functions are both defined as squared-error loss of the feature representations of the corresponding images [4]. They can be defined as

$$J = \alpha J_{content}(C, G) + \beta J_{style}(S, G) \quad (1)$$

Where α and β are hyperparameters which should be tuned by us, S is the style image, C is the content image and finally G is the output (resulting) image for the cost function. The most efficient algorithm to optimize the feature weights such that the target image is an acceptable blend of the style image into the content image is running gradient descent

on the total loss function. We will use the Adam Optimizer in our implementation as it is complex enough and fast. Firstly, we are assigning G resulting image randomly and then with the help of the update rule given below, parameters are going to be tuned.

$$G^{(t+1)} = G^{(t)} + \nabla G(t) \tag{2}$$

Generally, the content loss can be determined from the specified content layer of the model. Current values of the content image ($a^{[l](C)}$) and target image ($a^{[l](G)}$) from this layer are extracted and their difference is squared. Content loss can be formulized as:

$$J_{content}(C, G) = \frac{(a^{[l](C)} - a^{[l](G)})^2}{2} \tag{3}$$

Although the procedure is very similar for the style loss, two factors should be accounted for during the style loss calculation. First one is that generally there are more than one style layers and the second one is that gram matrices should be used for the style loss calculation. Calculation of the gram matrix for each style layer is pretty straightforward, multiplying the tensor with its transpose.

$$Gram_{kk'}^{[l](G)} = \sum_{i=1}^{n_H^{(l)}} \sum_{j=1}^{n_W^{(l)}} a_{i,j,k}^{[l](G)} a_{i,j,k'}^{[l](G)} \tag{4}$$

$$Gram_{kk'}^{[l](S)} = \sum_{i=1}^{n_H^{(l)}} \sum_{j=1}^{n_W^{(l)}} a_{i,j,k}^{[l](S)} a_{i,j,k'}^{[l](S)} \tag{5}$$

Where S and G stands for style and output images; i and j are the indices for width and height of the image.

The gradient descent hyperparameters “learning rate” and “step count” also need to be determined during the implementation of the project. Hyperparameter tuning will be done by running several tests with different parameter values.

As a final note, the transformations between image, tensor, and NumPy array with proper normalizations should be practiced properly as different parts of the implementation require different formats and normalizations.

3. What Is Done

Before starting to implement transfer learning with CNN, we learned the theoretical and practical parts of neural networks from perceptron to Convolutional Neural Networks. Moreover, we had to learn the implementation ways with PyTorch. As most of us had no prior knowledge about Deep Learning, the learning procedure took some time.

Since transfer learning implementation requires feature extraction from different layers, we learned the fundamentals of convolutional, pooling, and fully connected layers. We have searched for the optimal layers to extract content and style from the inputted image [3]. The article we found claims that the best style transfer with VGG 19 pre-trained model can be done using 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1', and 'conv5_1' layers of VGG 19 as style extraction layers and 'conv4_2' layers as the content extraction layer [3]. We have implemented our style transfer project using these layers and got acceptable results.

To implement the algorithm, we have learned the reasoning behind the transfer learning paradigm. As we do not have weeks for the implementation to run and a huge dataset to train a complex architecture as VGG 19, using a pre-trained model to start with is the best approach. As we searched for the alternatives, we found that the best pre-trained models are VGG 19, VGG 16, and AlexNet. Although we have not tried AlexNet yet, AlexNet promises some hope. We have chosen the VGG 19 model as it is the most complex model in between and promises more accurate content and style extraction from several layers.

After researching the procedure of style transfer, we have realized that the actual training does not occur on the model but on the target (final) image. The optimizer, Adam Optimizer, does not run on the model weight but on the target image to match its features with the content features and its gram matrix with the style gram matrix. The end product of the optimization procedure is the target image itself. What this means is that every content and style image pair needs to be trained from scratch to obtain their target image.

We have started our implementation by calling the VGG 19 pre-trained model from PyTorch. Then, we have freezed the weights of the feature parameters since we will not optimize them. After loading the content and style images, we have started the training procedure on the target image, which is currently empty. We have passed both content and style images to the model and extracted their features, immature outputs, from several different layers as mentioned before. To smoothen the optimization procedure, we have calculated the gram matrices of style features. At this point, we have calculated content and style losses from content features and style gram matrices and passed them

to the optimizer to minimize these losses on the target image. To visualize the procedure, we have printed the iterations with loss values and show premature target images during the training. We have also created a video by saving premature target images and combining them.

There are several hyperparameters of this implementation which we will search for the best target image. First of all, the implementation uses a learning rate for the Adam Optimizer as all CNNs. Changing the learning rate affects the target image drastically. Moreover, we have two weights for the content and style losses, their ratio determines the total loss value and so the final ratio of content and style in the target image. Finally, we have a variable deciding the number of iterations the optimizer will work on the target image. Increasing the variable boosts the style of the target image.

The following figures show an example run with our current implementation;

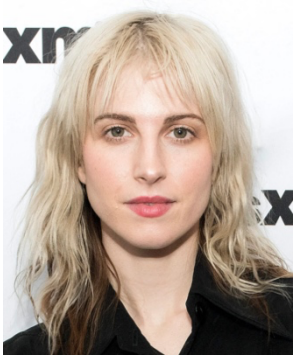

Figure 1: Content Image (Hayley Williams)

Figure 2: Style Image (Angle of Love)

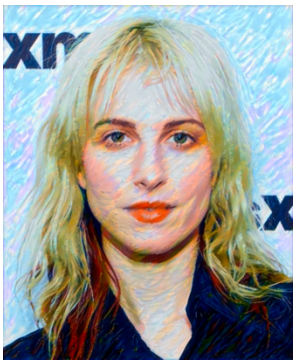

Figure 3: Target Image with 1500 iterations

Figure 4: Target Image with 2200 iterations

4. What Remains to Be Done

As mentioned earlier, Style Transfer is quite different than the traditional CNN approaches since it's training procedure and optimizer runs on the target image rather than on the model itself. This situation creates a huge problem in terms of runtime because different image pairs require a new and different training and so each new input pair has to wait for the entire training procedure to obtain the target image. Although this abnormality will be investigated further by us, it allowed us not have to work with the datasets immediately. In the first step, we have worked with two images, one for content and one for style. We will extend our implementation to work with the datasets mentioned in the Proposal Report. Moreover, the user should be able to specify the art movement for the style image. We will support these features with several datasets.

Different pre-trained models such as VGG19, VGG16, and AlexNet will be tested to see the best fit. Moreover, different layers for content and style extraction should be tested. The most promising one is to test the 'conv5_2' layer of VGG19 as the content extraction layer.

Finally, we will test different values for content weight, style weight, learning rate, and the number of steps (iterations). These are some of the hyperparameters our implementation has and should be tuned properly.

5. Work Division

Ömer Ünlüsoy: CNN and PyTorch Learning, Transfer Learning with VGG19, Image Extraction, Training Procedure, Video Creation, What Is Done and What Remains to Be Done sections of Progress Report

Doğa Tansel: PyTorch Learning, Introduction, Background Research, What Remains to be Done sections of Progress Report

Ata Berk Çakır: CNN and PyTorch Learning, Literature review to understand the mathematical background of the CNN, Background research part of Progress Report

Mustafa Hakan Kara: Convolutional Neural Networks theoretical learning, Literature Review

Ahmet Cemal Alıcıoğlu: CNN Learning, PyTorch Learning

6. References

[1] Y. Shih et al. “Style Transfer for Headshot Portrait.” Available: https://people.csail.mit.edu/yichangshih/portrait_web/2014_portrait_hires.pdf

[2] M. Lu, F. Xu, H. Zhao, A. Yao, Y. Chen and L. Zhang, "Exemplar-Based Portrait Style Transfer," in *IEEE Access*, vol. 6, pp. 58532-58542, 2018, doi: 10.1109/ACCESS.2018.2874203.

[3] 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), 2016, pp. 2414-2423, doi: 10.1109/CVPR.2016.265.

[4] L. A. Gatys, A. S. Ecker and M. Bethge, “A Neural Algorithm of Artistic Style,” arXiv:1508.06576