Bilkent University
Department of Computer Engineering

CS-464
Introduction to Machine Learning

Progress Report

Stylist

Group Name:	SKYNET
Group Number:	Section 2, Group 5
Group Members:	Sait Aktürk 21501734
	Kerem Ayöz 21501569
	Cansu Yıldırım 21502891
	Metehan Kaya 21401258
	Alper Kılıçaslan 21502103

1. Introduction

Creating art piece with computers always attracts people. In this project we want to study one of the techniques to do that which is Style Transfer. In this process, algorithm takes 2 input images; 1 content image and 1 style image. Then the algorithm tries to transfer the style of the style image to content image. Here style means the high-level features of the style image and content means the low-level features of the content image. Our goal is to combine that features into one single image.

There are several techniques and implementations that achieve style transfer between images. We will use dataset as COCO2014 for perceptual losses for style transfer [1]. For, Gatsy's techniques, we will use imagenet-pretrained VGG16 CNN model [2]. Both models need a style image to transfer style.

We develop this project with Python programming language and use various different Machine Learning libraries such as Tensorflow and Keras. We mainly use Google Colab and Anaconda development environments in that project. In that progress report, two different approaches are examined and results are shown.

2. Background Information

Style Transfer algorithms are various, some of them are trained for each different style [4] while some of them are able to use different style images without any explicit training [3]. Both of these methods have advantages and disadvantages. Universal Style Transfer algorithms [3] have shorter training time but longer running time whereas other methods [4] have longer training time however can produce output image instantly.

Main idea in both of the methods that we used is transferring high level features of style image which are taken from a pre-trained

Convolutional Neural Network Image Classifier model into the low level features of content image which is also taken from the same model. These features are taken from different layers of the same classifier. Other various methods [5] and [6] use different neural network architectures to obtain these image features.

3. Progress

In the methods that we implemented, style transfer is done in two different ways. In the first one [3], there is no need for explicit training for different style images and algorithm uses pre-trained model to solve optimization problem. In second method [4], a Convolutional Neural Network (CNN) model needs to be trained for each style. First method has longer average runtime than the second one however, it does not require explicit training. We tried both of the methods and we saw that both of the models give similar results in terms of output image however output image generation latency and training time differ these models. We will describe these two methods, their performances and sample outputs.

3.1. Gatsy's Method

In the article [3], method proposed by Gatsy uses a universal style transfer algorithm. In that algorithm, we have 2 input images; 1 content image and 1 style image.

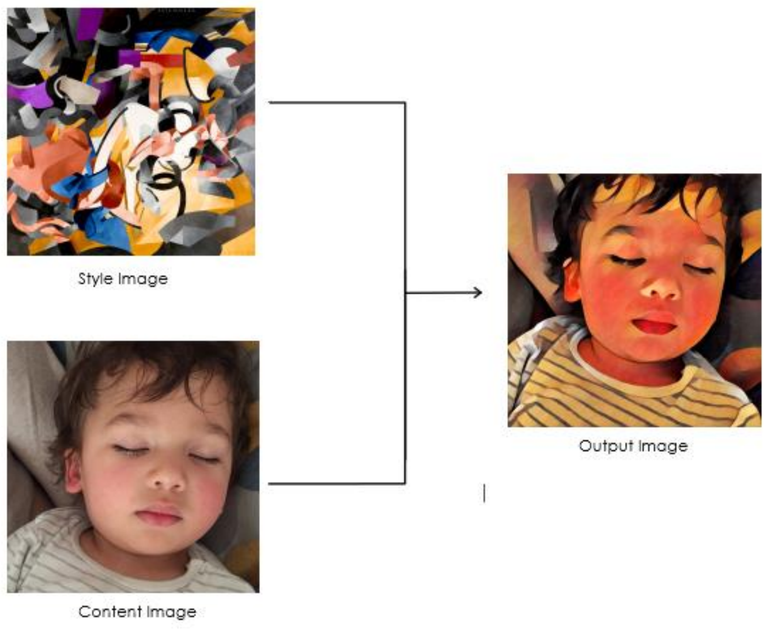


Fig 1. Example of input/output combination

In this method, we downloaded a pre-trained CNN model [2] "VGG16" which won the ImageNet competition. It is a 16-layer network and architecture are as following:

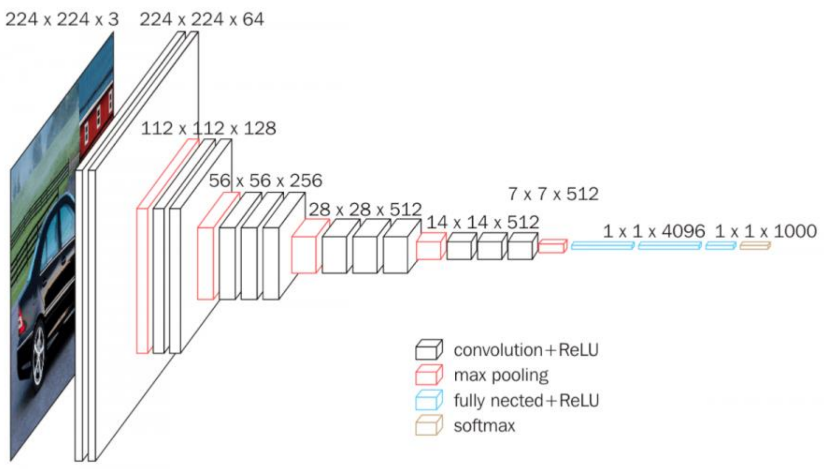


Fig 2. Architecture of VGG16 CNN

We used that pre-trained model to extract the features of the content image and style image. First layers of that model represent the low level features of the input image such as edges, last layers of the model represent high level features such as objects, colors. We feed both of these images into network. After that, we initialized an output image as randomly and tried to calculate a loss function to optimize with respect to that output image. The output image is created by finding an image which matches the content of the content image and the style of the respective style image.

3.1.1. Loss Function

Loss function contains two main parts; content loss and style loss. We tried to optimize the weighted sum of these loss functions and find the best output image that minimizes the loss.

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \left(\alpha \mathcal{L}_{\text{content}}(\mathbf{c}, \mathbf{x}) + \beta \mathcal{L}_{\text{style}}(\mathbf{s}, \mathbf{x}) \right)$$

Here α and β are weights of the losses and x is the output image that we tried to find. c and s represent content and style.

The content loss is the Euclidean distance between feature vectors, which are obtained from the CNN model, of the content and output images. For the content loss, we follow Gatsy et al. (2015) and extract the content feature from `block2_conv2` layer. Selecting different layers for content loss gives different results in the output however, we followed the suggested layer's features.

The style loss is calculated by something called 'Gram Matrix'. Entries of this matrix represents the covariances of the given set of features. Gram Matrix enables us to obtain the information about features that are activated together. With that information, we capture the style of an image regardless of the arrangement of these features in style image.

This is the core of our algorithm, we become able to capture the style independent from the content of the style image by calculating the Gram Matrix. We calculated the Gram Matrix of the style image and output image and take their squared distance to calculate the style loss.

3.1.2. Minimizing the Loss Function

Then we came to the point where we needed to optimize the loss function that we created. We chose to use something called ‘L-BFGS algorithm’, which is a quasi-Newton algorithm that converges faster than the standard gradient descent. After certain number of iterations, output image is produced. Here we do not train an explicit model. Instead, we optimized a loss function to obtain output image. However, this optimization needs to be done for each content and style image. It takes 25 seconds per iteration on the average with Nvidia Tesla K80 GPU which is provided by Google Colab. Thus, getting the output approximately takes 4-5 minutes with that algorithm if we use 10 iterations. In the next method, we tried to optimize that running time by a great factor, however we sacrificed the training time.

3.2. Johnson’s Method

In 2016, Johnson et al. proposed new style transfer algorithm [4] that is up to “~1060x” faster than Gatsy et al. proposed algorithm [3]. They changed loss function using per-pixel loss function that compares two images based on their individual pixel intensities or values to perceptual loss functions that compares two images based on high-level features obtained from pretrained VGG16 model.

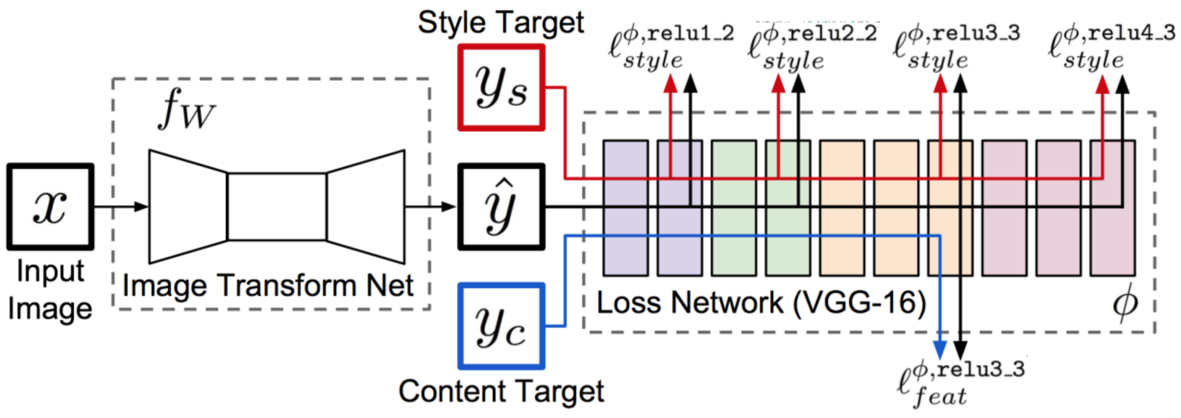

Style Target
$\ell_{style}^{\phi,relu1\_2}$
$\ell_{style}^{\phi,relu2\_2}$
$\ell_{style}^{\phi,relu3\_3}$
$\ell_{style}^{\phi,relu4\_3}$
y_s
f_W
x
$\hat{y}$
Input Image
Image Transform Net
y_c
Loss Network (VGG-16)
ϕ
Content Target
$\ell_{feat}^{\phi,relu3\_3}$

Fig 3. System overview of Johnson's Method

The algorithm consists of two networks that are image transformation network and loss network. Image Transformation Network is a convolutional neural network that has residual blocks and fractionally & strided downsampling and upsampling. The input and output image have the same shape, the training is basically done for this part of the algorithm with reconstruction perceptual loss. Loss Network is also a convolutional neural network that the paper uses VGG16 pretrained model on imagenet dataset [4].

There are two main perceptual losses that the Image Transformation Network is training according to these loss functions. First loss function is feature reconstruction loss that finds the Euclidean distance between feature representation obtained from pretrained VGG16 model intermediate layers. The feature reconstruction loss boosts the output image to be perceptually similar to the target image. The style reconstruction loss uses the frobenius norm of the difference between Gram matrices that is performed in several intermediate layers of VGG16 model to get high level features of the output image. The style reconstructıon loss saves the stylistic representations from the target images, although it does not preserve the target image's spatial structure.

The algorithm is trained 4-6 hours Nvidia Tesla K80 which is provided by Google Colab. The main difference of this algorithm to Gatsy et al.

algorithm's speed in terms of reconstructing the image and perceptual loss rather than per-pixel loss. The trained model could be used for other images without retraining that the speed difference comes from that part. Two algorithms have similar objectives and qualitative results as certain optimization over image.

4. Result and Discussion

We tried different content and style images with our algorithms with different parameters to understand and analyze the methods proposed.

4.1. Gatsy's Method

We obtained the following total error plot while minimizing our loss function with L-BFGS algorithm.

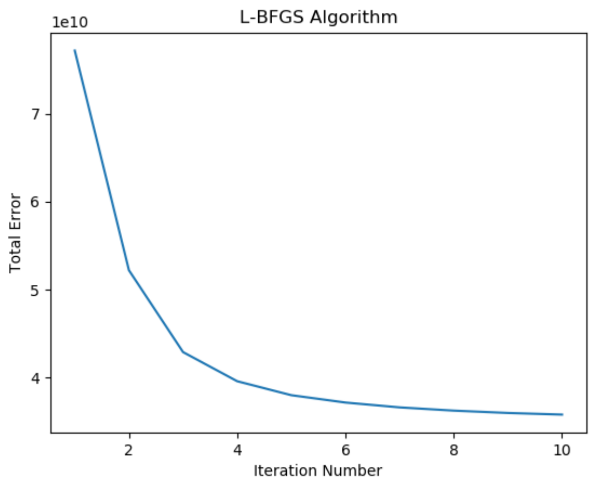
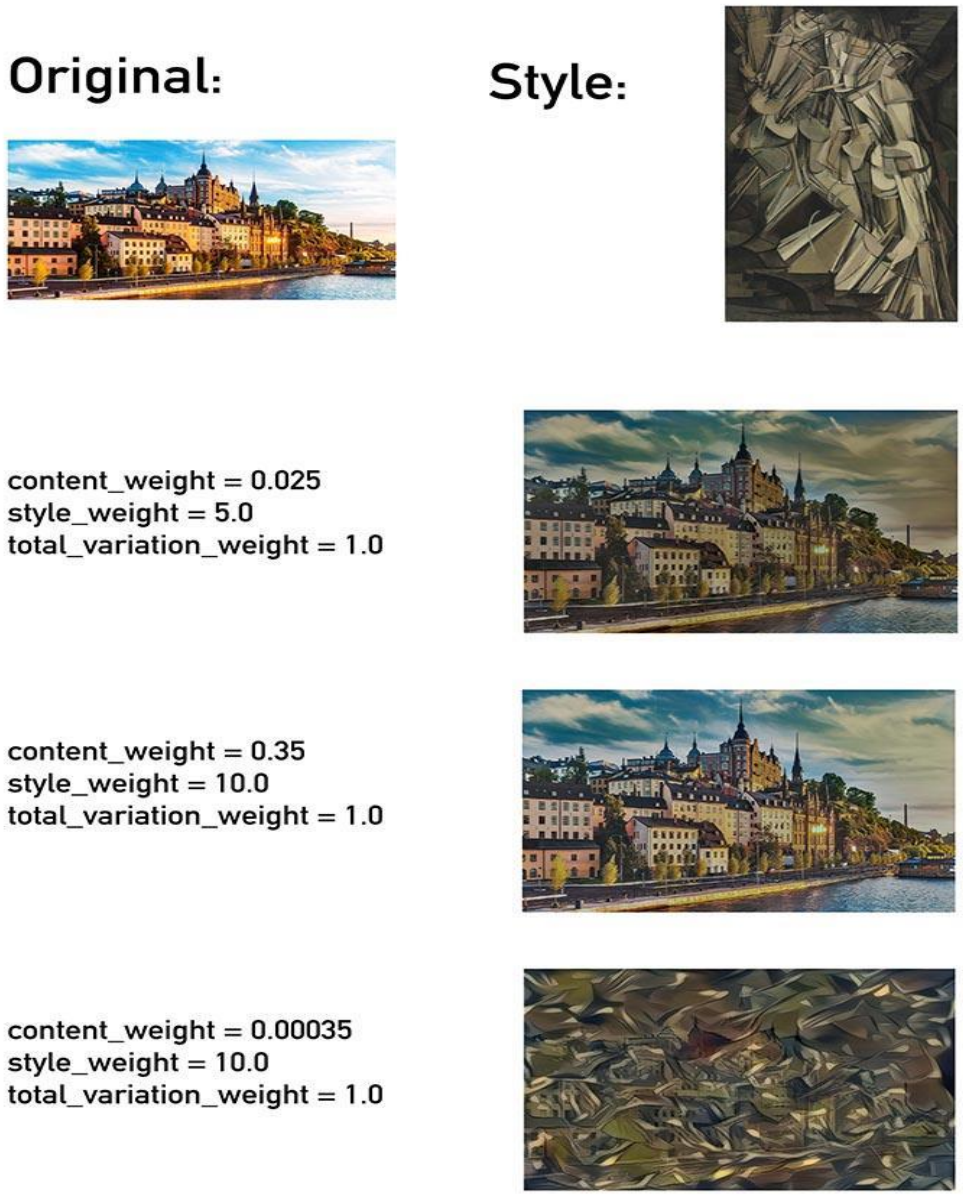


Fig 4. Total error in each iteration

Here we can see that 10 iterations are enough to decrease the error to a reasonable amount compared to the cold-start error rate. Also, we observed that after 10 iterations, changes in the output image become unnoticeable, so we decided to do 10 iterations in each execution. We run Gatsy's method with different loss weights to see difference. Also, we changed the layer that we took the features of the content image and observed that it produces different outputs, but we decided 'block2_conv2' gives better results than 'block4_conv2' layer. Here increasing style weight makes an image to be more similar to style image. While increasing the content weight makes it to be more similar to content image.

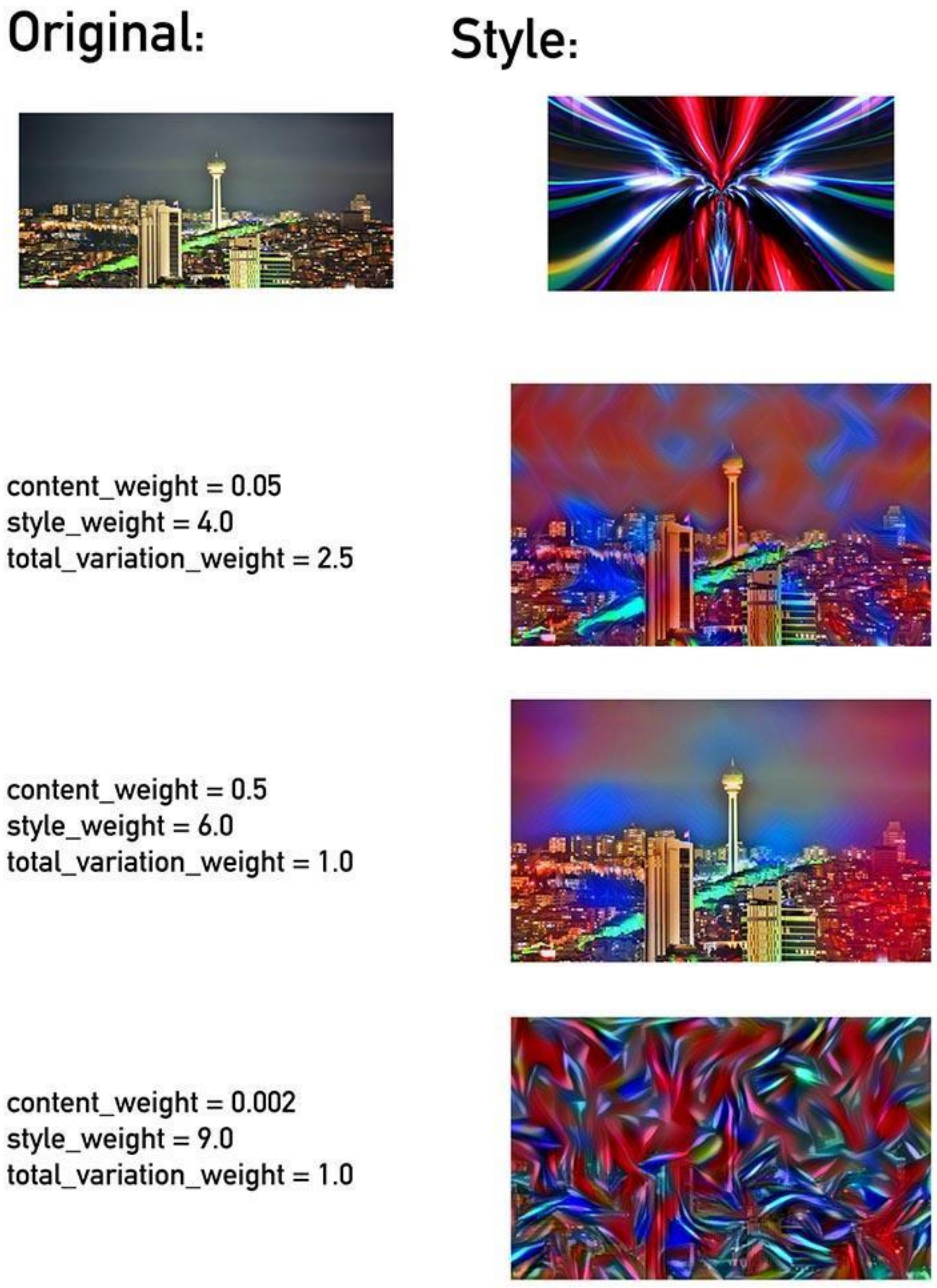


content_weight = 0.025
style_weight = 5.0
total_variation_weight = 1.0

content_weight = 0.35
style_weight = 10.0
total_variation_weight = 1.0

content_weight = 0.00035
style_weight = 10.0
total_variation_weight = 1.0

Fig 5. Example execution of Gatsy's method with different weights of losses



content_weight = 0.05
style_weight = 4.0
total_variation_weight = 2.5

content_weight = 0.5
style_weight = 6.0
total_variation_weight = 1.0

content_weight = 0.002
style_weight = 9.0
total_variation_weight = 1.0

Fig 6. Example execution of Gatsy's method with different weights of losses

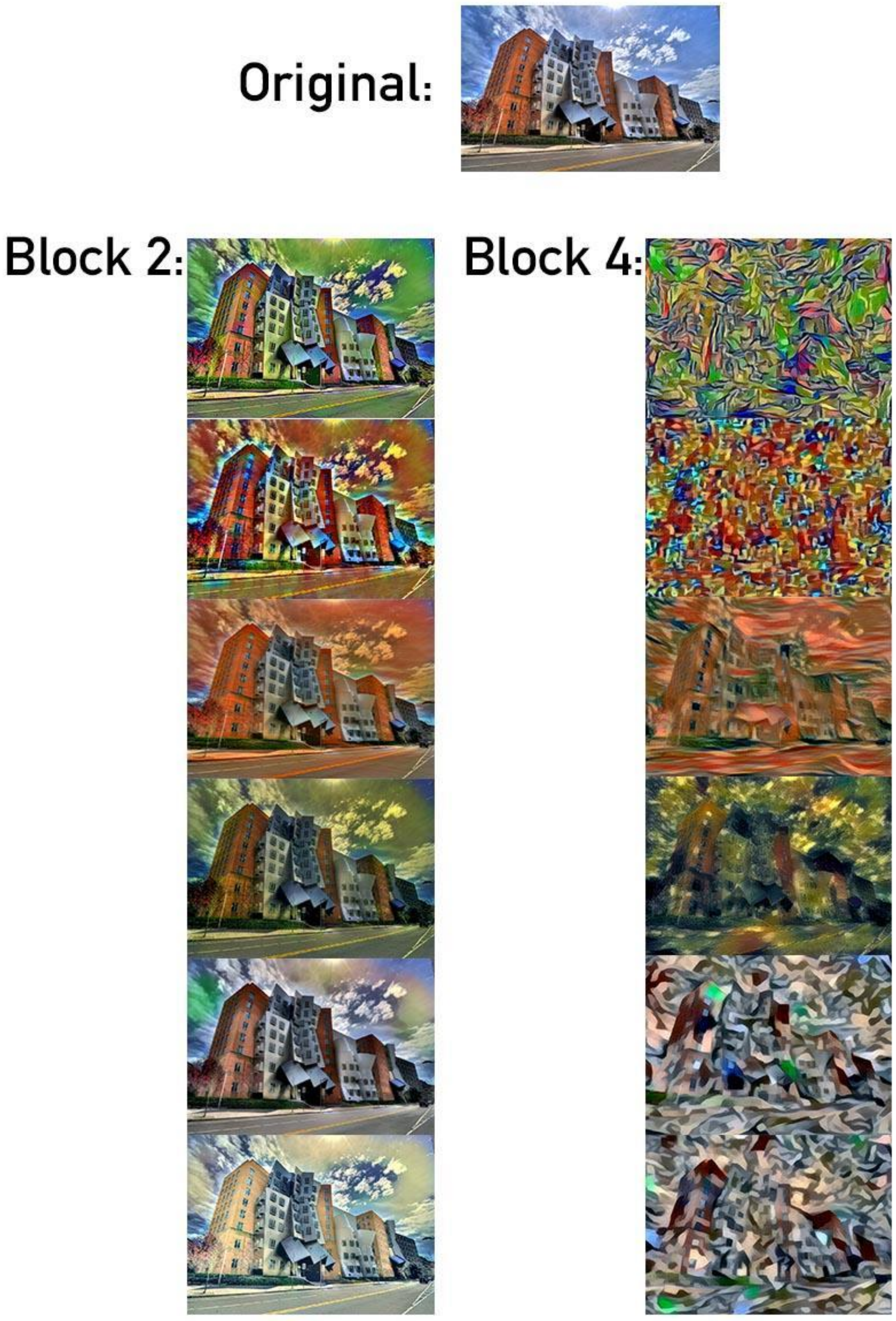
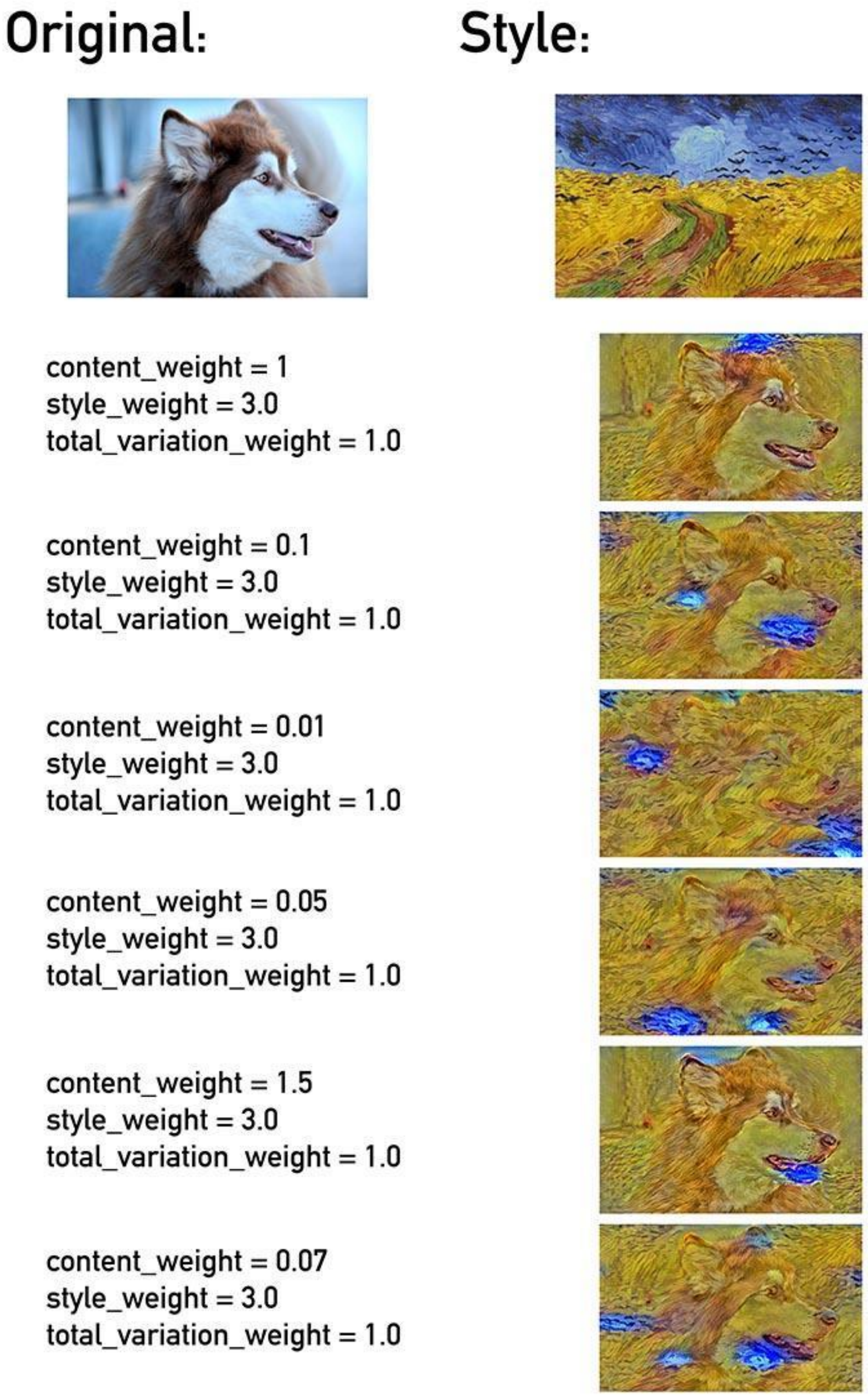


Fig 7. Example execution of Gatsy's method with different content feature layers



content_weight = 1
style_weight = 3.0
total_variation_weight = 1.0

content_weight = 0.1
style_weight = 3.0
total_variation_weight = 1.0

content_weight = 0.01
style_weight = 3.0
total_variation_weight = 1.0

content_weight = 0.05
style_weight = 3.0
total_variation_weight = 1.0

content_weight = 1.5
style_weight = 3.0
total_variation_weight = 1.0

content_weight = 0.07
style_weight = 3.0
total_variation_weight = 1.0

Fig 8. Example execution of Gatsy's method with different content weights

4.2. Johnson's Method

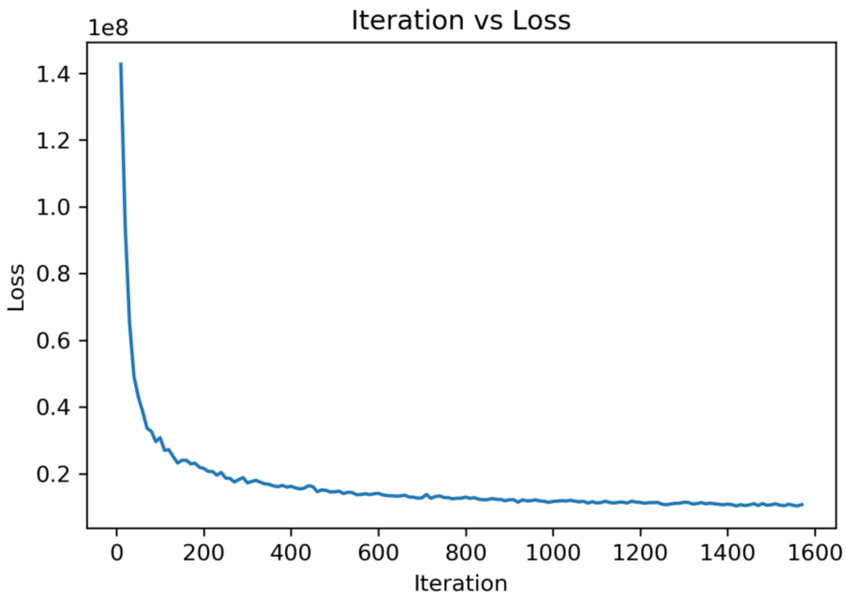


Fig 9. Total loss in each iteration

This is the total loss vs iteration plot of Johnson's method. After some hyperparameters tuning that batch size 16, learning rate 0.0075, content weight 10 and style weight 75. We got this plot via running 1600 iterations over training dataset to get the network model for testing images. We also plotted the style loss vs iteration plot as following:

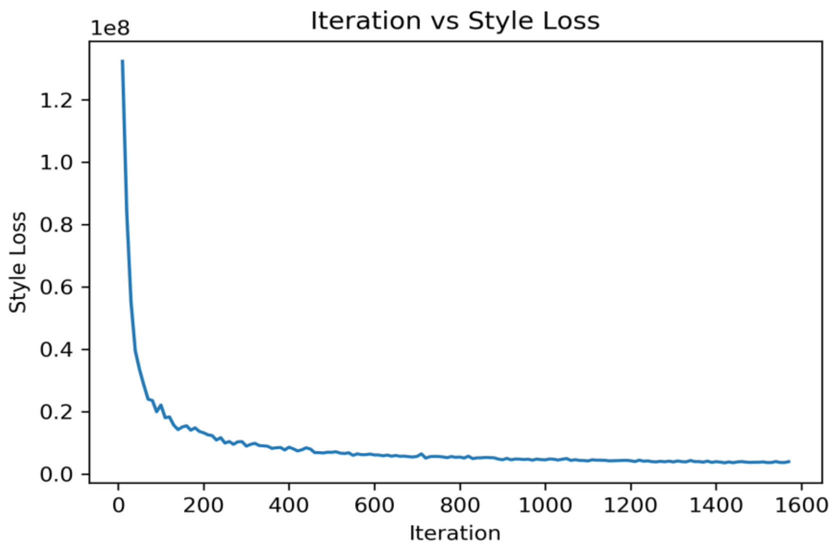


Fig 10. Total loss in each iteration

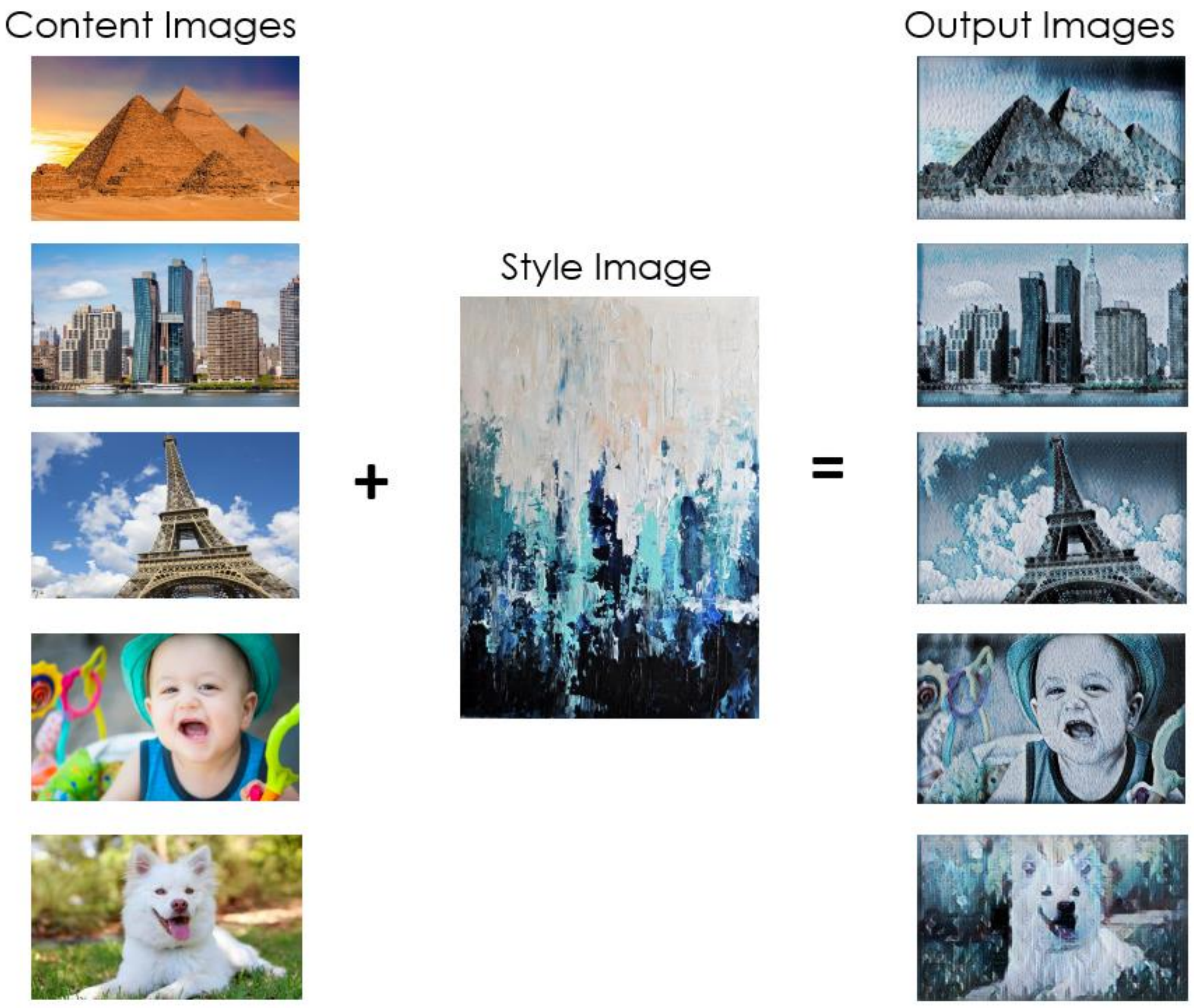


Fig 11. Example input-output pairs obtained with Johnson's method

The pretrained style transfer model is tested with given content images in above figure. Each content images have different image sizes for testing. The evaluation time is approximately 75-100 ms for each image on Nvidia Tesla K80.

5. Future Work

We will use other algorithms that Deep Photo Style Transfer [5] and Universal Style Transfer [6] for testing. We will evaluate these algorithms and find differences in terms of training time, evaluation time and aesthetic and other performance metrics. After finding the best algorithm for content aware style transfer, we will use that style transfer algorithm for real time fashion style change that we will build DeepLabV3+ [7] which is going to be used for semantic segmentation to get pixels specific fashion parts like pants, jackets, t-shirts, etc. We will use ModaNet [8] dataset with polygon annotations for training DeepLabV3+. The style transfer will be applied only specific fashion parts that we will get from DeepLabV3+ model. We will also try to apply style transfer to create special effects for human body parts to test robustness of the style transfer algorithm.

6. Work Division

Sait Aktürk - Built and optimized the Johnson's method and obtained the plots of the Johnson's method.

Cansu Yıldırım - Optimized the Johnson's method parameters and obtained the results of the Johnson's method.

Kerem Ayöz - Built and optimized the Gatsy's method. Obtained the plots of the Gatsy's method.

Metehan Kaya **-** Optimized the Gatsy's method parameters. Obtained the outputs of the Gatsy's method.

Alper Kılıçaslan **-** Optimized the Gatsy's method parameters. Obtained the outputs of the Gatsy's method.

7. References

[1] "Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Doll´ar, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV. 2014.

[2] K. Simonyan and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR abs/1409.1556 (2014)

[3] L. Gatys, A. Ecker, and M. Bethge, "A Neural Algorithm of Artistic Style," Journal of Vision, vol. 16, no. 12, 2016.

[4] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," *Computer Vision – ECCV 2016 Lecture Notes in Computer Science*, pp. 694–711, 2016.

[5] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep Photo Style Transfer," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[6] Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Universal style transfer via feature transforms. In: Advances in Neural Information Processing Systems. 2017

[7] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. arXiv preprint arXiv:1802.02611, 2018

[8] S. Zheng, F. Yang, M. H. Kiapour, and R. Piramuthu. Modanet: A large-scale street fashion dataset with polygon annotations. In MM, 2018