COL870 Deep Learning for Computer Vision **Semester 1 2022-23**

Neural Style Transfer

Instructor: Anurag Mittal Harshil Vagadia, Rishi Shah

1 Introduction

Neural Style Transfer tries to leverage deep networks to create artistic images. Given a content image and a style image, we try to transfer the style to the content image. The algorithm starts with the content image and iteratively transfers the styles to it. We provide detailed experimentation on various hyperparameters of the algorithm that will hopefully provide valuable insights on finetuning the styled images. All the results in this report are qualitative in nature.

2 Approach

This algorithm relies on pre-trained deep CNNs to extract features from images. The main paper uses VGG19 model trained on classification task. We experiment with different architectures.

The algorithm is iterative i.e. the quality of output image improves with each run. There are two losses: Content Loss and Style Loss. The overall loss is the weighted combination of both.

$$\mathcal{L} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$$

3 Content Loss

The content loss tries to keep the generate image as close to the content image. Since CNNs are good at extracting important features from an image, the content loss at a deeper layer in the pretrained network. The features at deeper layer will only contain the important aspects of an image and ignore the styling part. Content loss is simply the MSE loss between deep features of content and generated images.

$$\mathcal{L}_{content} = \frac{1}{2}\Sigma_{i,j}(A^l_{ij}(g) - A^l_{ij}(c))^2$$

Here the loss is taken for l-th layer and $A^l(I)$ represents the features of the l-th layer.

4 Style Loss

Style loss is generally taken over multiple layers. For each layer, it is the MSE between the gram matrices of the style and generate images.

$$\mathcal{L}_{style} = \Sigma_l w^l \mathcal{L}^l_{style}$$
$$\mathcal{L}^l_{style} = \Sigma_{ij}(G^l_{ij}(s) - G^l_{ij}(g))^2$$
$$G^l_{ij}(I) = \Sigma_k A^l_{ik}(I)A^l_{jk}(I)$$

The Gram matrix essentially captures the distribution of features of a set of feature maps in a given layer. The style loss essentially matches the distribution of features between the two images.

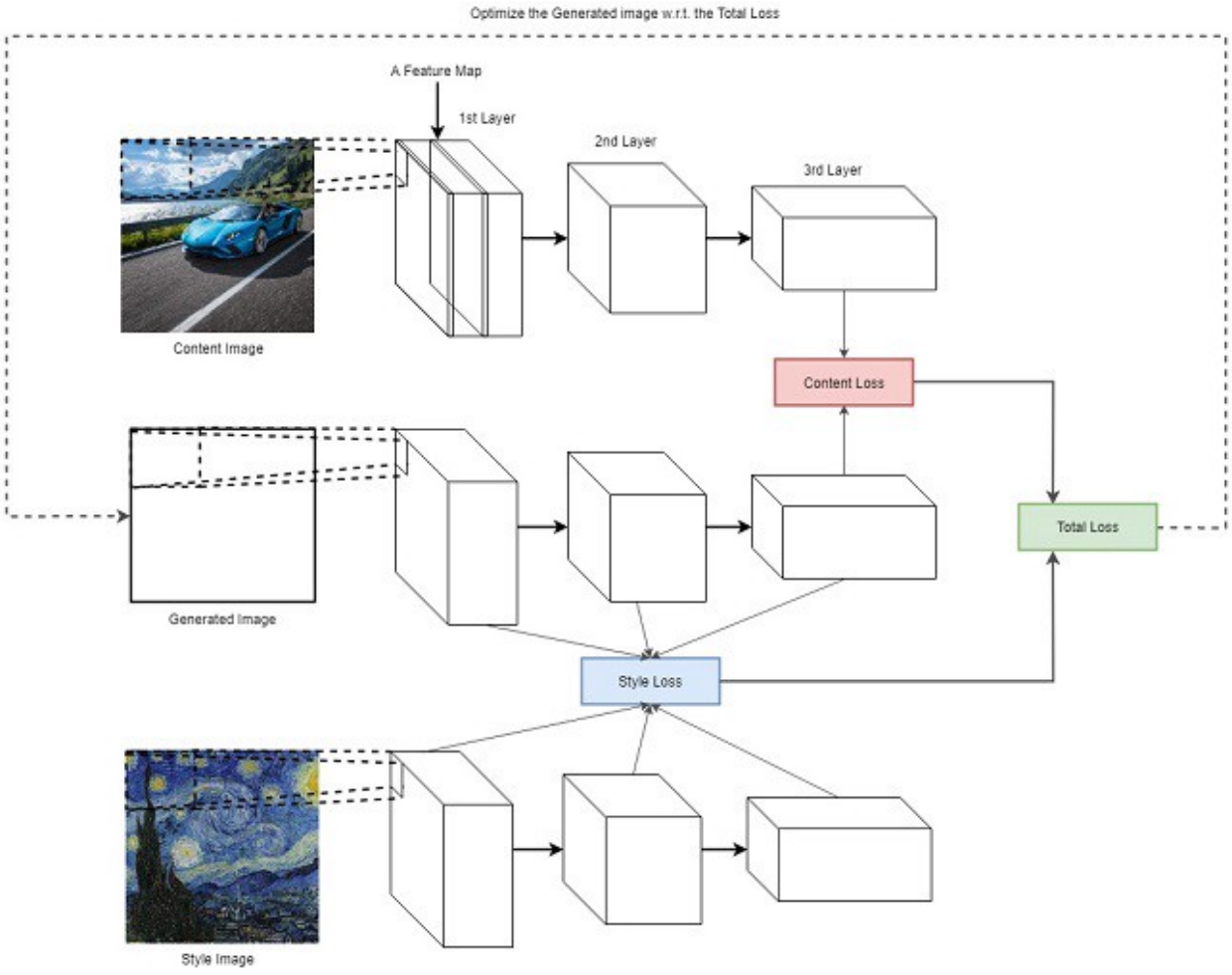

Figure 1: Overall Architecture

5 Experiments

5.1 Number of Epochs

Table 1: Variation over number of epochs (starting from content image.

We start with the content image and gradually optimise the total loss at each epoch. As we can see from the diagram, the styles gradually gets incorporated into the image while preserving the content.

Even if we start from a random (noisy) image we still get the styled image. Starting from content image simply reduces the iterations required.

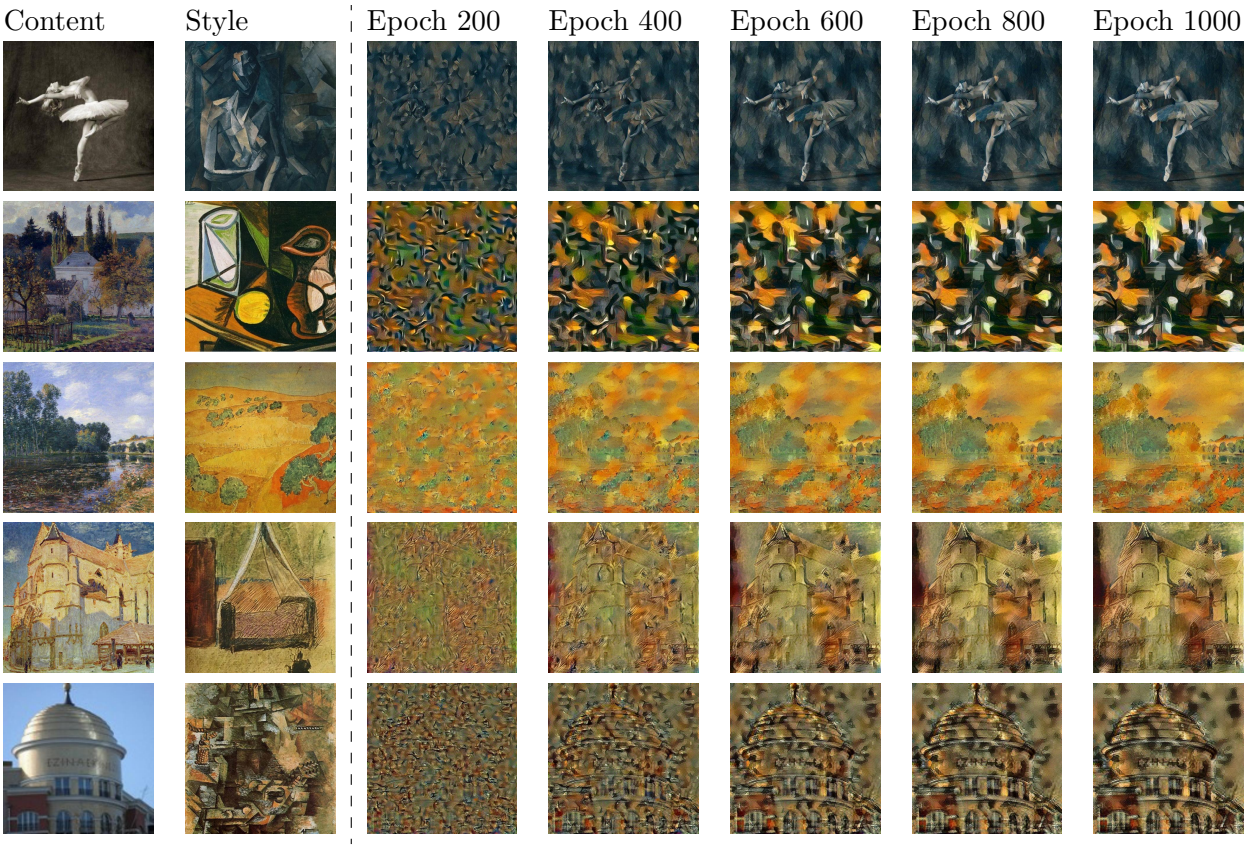

Table 2: Variation over number of epochs (starting from random image).

These experiments shows that the loss function is able to capture the style transfer goal accurately.

5.2 Weights of Content and Style Loss

In this experiment, we change the relative weights of content and style loss. As expected, the resulting image has increased styling as we increase the relative weight of style loss.

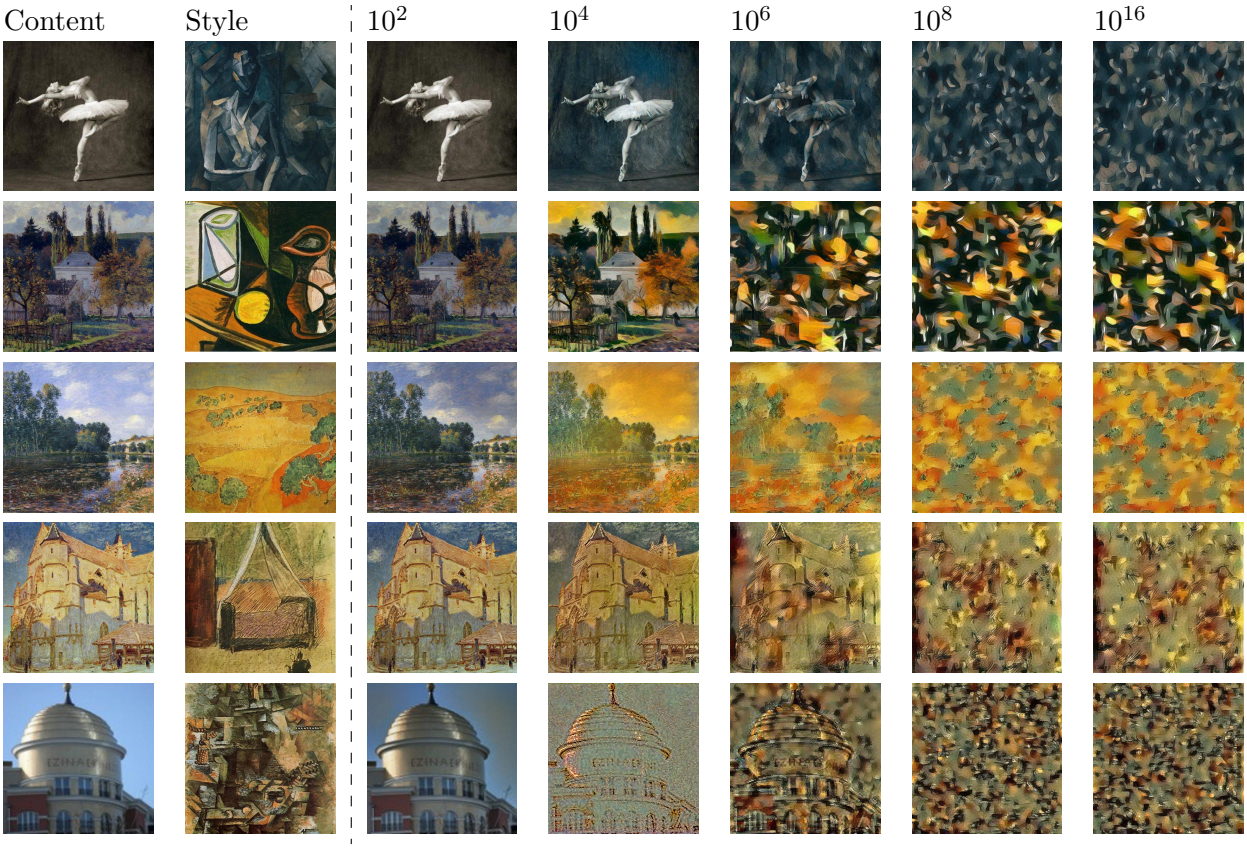

Table 3: Variation over relative weight of style loss w.r.t content loss.

5.3 Content Loss Layers

In this experiment we change the layers over which we find that the output is highly dependant on the layer chosen. Some layers give nicely styled image while others give noisy images. We hypothesise that some layers capture the content well while others acts as buffers for the following layers.

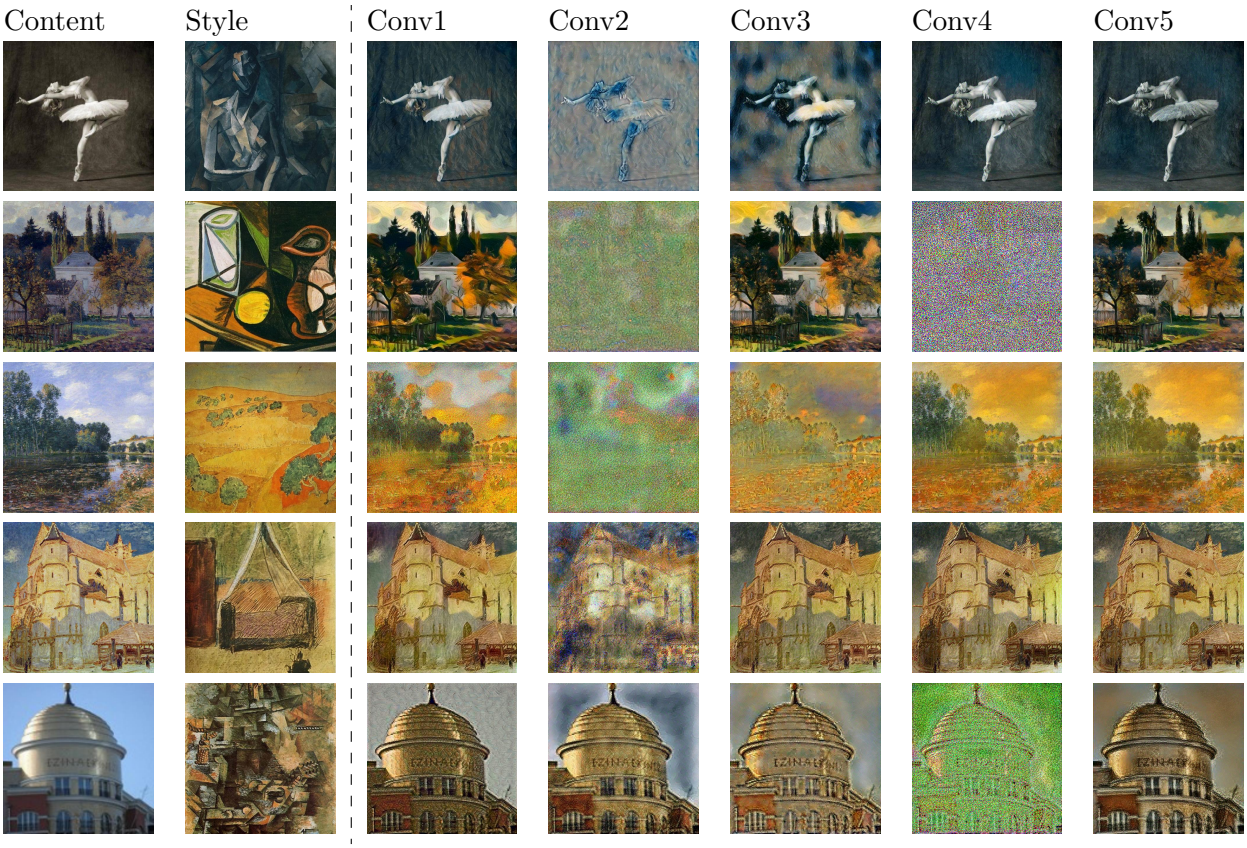

Table 4: Variation over content loss layers.

5.4 Style Loss Layers

In this experiment we change the layers over which we calculate the style loss. We observe that style is captured better in the earlier layers. Typically style loss is taken over few consecutive layers.

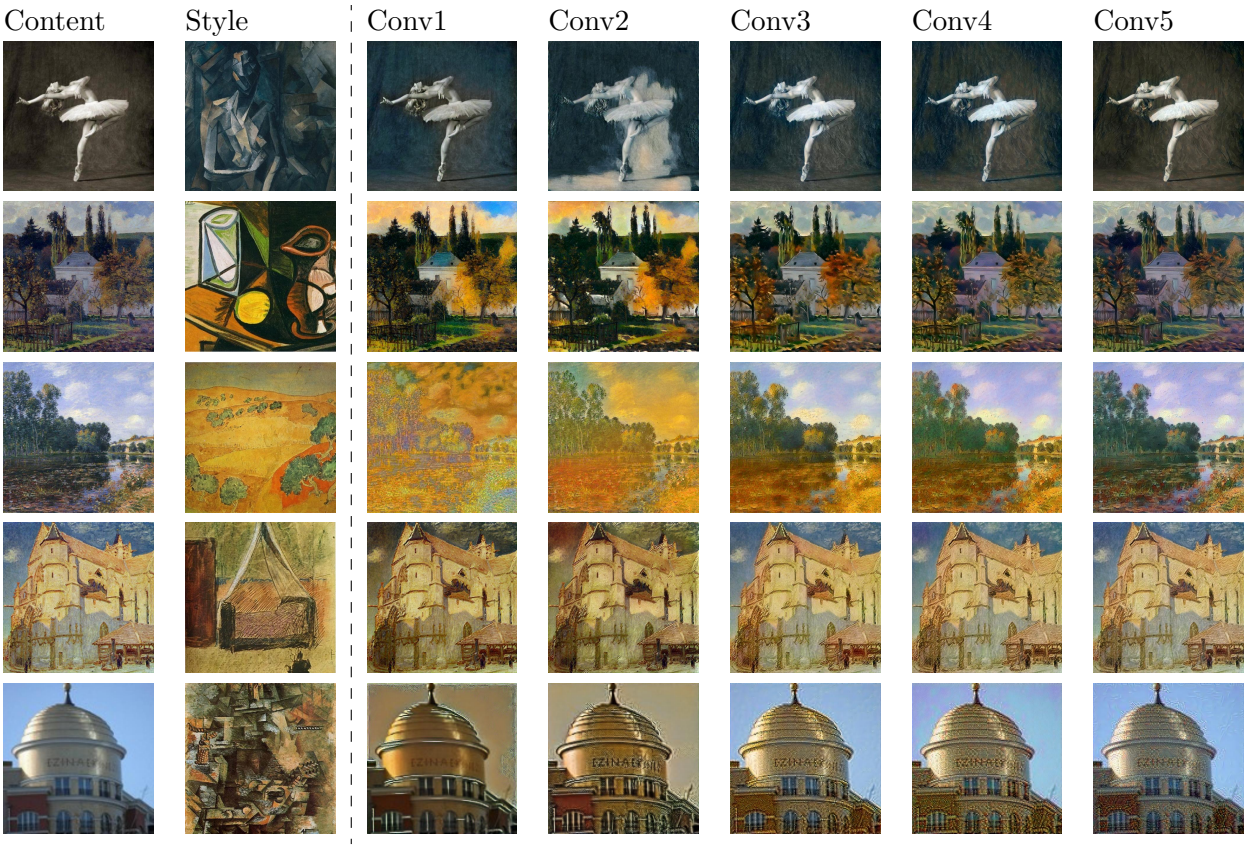

Table 5: Variation over style loss layers.

5.5 Different Model Architectures

In this experiment, we change the pretrained CNN model. We try ResNet and GoogleNet along with VGGNet (originally). We find that each model is able to create styled images. However, there are subtle differences in each model.

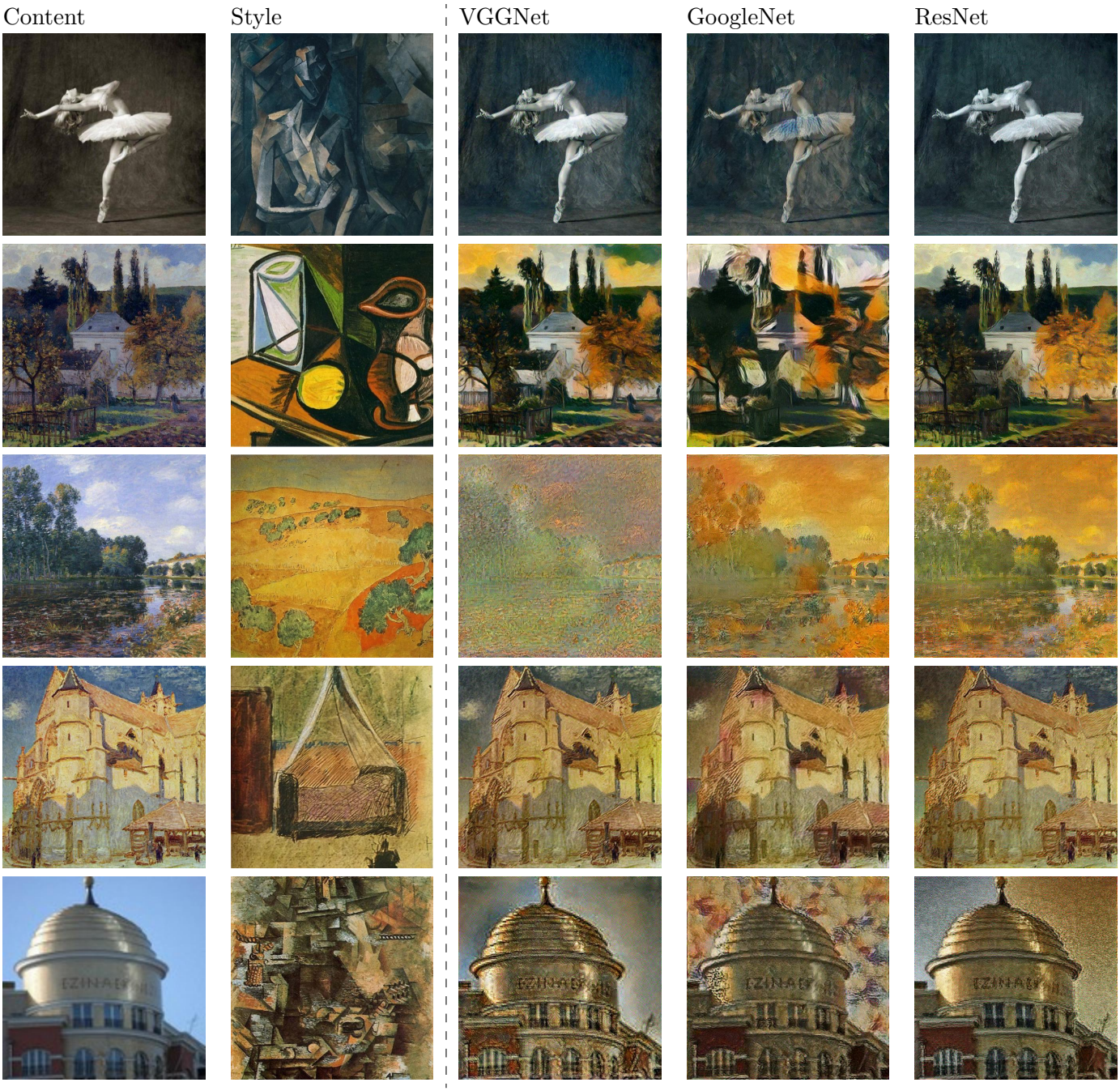

Table 6: Variation over different models.

5.6 Code Repository

It contains all the details to run the code. Change the makefile to run the different experiments.

5.7 References

1. A Neural Algorithm for Artistic Style, Gatys et. al. (https://doi.org/10.48550/arXiv.1508.06576)

2. Intuitive Guide to Neural Style Transfer, Towards Data Science