Monkeying Around: A Comparative Analysis of VGG19 and DenseNet201 on Neural Style Transfer

Africa, Ong Yiu, Pastor

Dececmber 2023

1 Introduction

If you've ever seen the anime filters that went viral on TikTok in the last year, you were probably seeing the result of Neural Style Transfer (NST), the process of using deep learning techniques to impose the artistic style of one image on the content of another. The inception of this field can be traced back to the seminal work of Gatys et al. [2], which introduced a novel approach based on convolutional neural networks (CNNs) which separate the latent representations of content and style. Since then, NST has accumulated significant steam due to its potential applications in digital art, image enhancement, and creative content generation.

In this paper, we delve into NST, discussing its architecture, assessing the quantitative performance of two base models (VGG19 and DenseNet201), and linking them to qualitative comparisons. We will focus on the basic methodology outlined in [2] to narrow our scope, as opposed to covering all the intricacies and nuances in state-of-the-art approaches.

For better comparison of images, all images used in this paper are uploaded in this Google Drive.

2 Preliminaries

Before we begin our discussion on NST, it would be instructive to first discuss the architecture of the two base models that we will be using in this paper - VGG19 and DenseNet201. Both models are CNNs that have been trained on the ImageNet dataset to classify images and are available with their pre-trained weights in the PyTorch library [6]. The ImageNet dataset "spans $1,000$ object classes and contains $1,281,167$ training images, $50,000$ validation images, and $100,000$ test images" [1].

2.1 VGG19

The VGG model is based on a paper written by the Visual Geometry Group (VGG) in 2015 titled "Very Deep Convolutional Networks for Large-Scale Image Recognition" [7]. There are several version of the VGG model, but for the purposes of this paper, we will be focusing on the VGG19 model.

The VGG19 model is a 19-layer CNN (16 convolutional layers and 3 fully-connected layers). Since we are only interested in the feature maps of the images, we ignore the fully-connected layers which are used for image classification. In particular, the convolutional layers of the VGG19 model consist of five groups of convolutional layers separated by a maximum pooling layer (see Fig. 1). The five groups of layers have 2, 2, 4, 4, and 4 convolutional layers respectively.

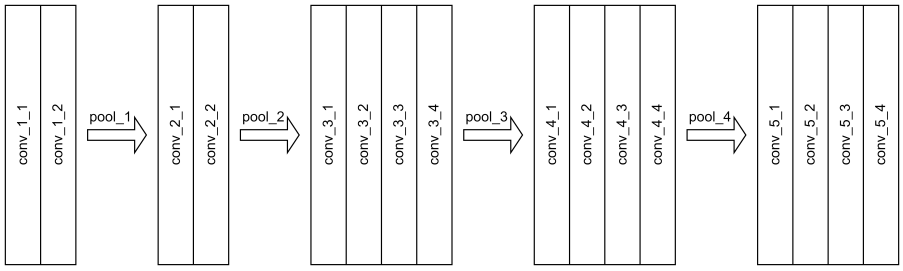


Figure 1: VGG19 model architecture with the classification layers removed

2.2 DenseNet201

The DenseNet model was proposed in 2018 in the paper "Densely Connected Convolutional Networks" by Huang et al. [4]. Like the VGG model, there are several versions of this model, but for this paper, we focused on the DenseNet201 model.

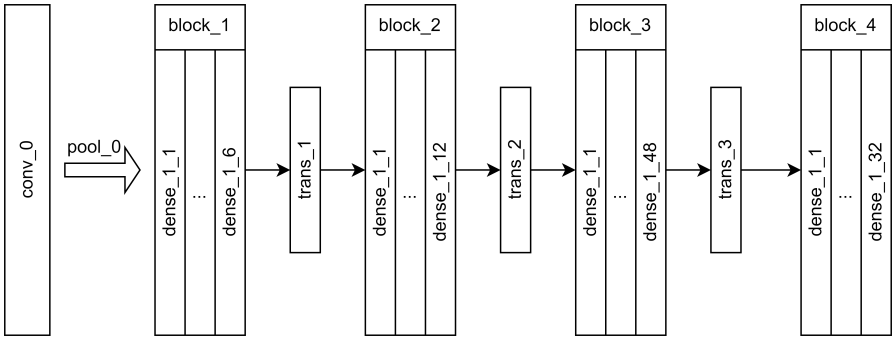


Figure 2: DenseNet201 model architecture with the classification layers removed

The DenseNet201 model is a 201-layer CNN (200 convolutional layers and 1 fully-connected layer). Similar to VGG19, since we are only interested in the feature maps of the images, we ignore the fully-connected layer which is used for image classification. The architecture of the DenseNet201 groups these convolutional layers into 4 dense blocks and transition layers in between those blocks. Each dense block in sequence consists of a different number of dense layers (6, 12, 48, and 32 layers respectively), each of which are further composed of 2 convolutional layers. Between these blocks are the transition layers, that each consist of a single convolutional layer.

As described in [4], these dense layers within the same blocks are unique in comparison to other convolutional networks due to the the presence of skip connections between said layers. Within blocks, each dense layer is directly connected to all subsequent layers, such that the kth layer receives the feature maps of all preceding layers concatenated together as input.

3 Neural Style Transfer Architecture

The basic task of NST is to take two images (one we will call the content image, and the other we will call the style image), and output a third image with the content of the content image and the style of the style image. This is not obvious or straightforward, as what constitutes "content" and what constitutes "style" is difficult to articulate. Nonetheless, if you have ever listened to a band cover another band's song, you will have a sense for what this means. It has the same lyrics, but a different voice (same content, but a different style). In fact, the central conceit of the original paper by Gatys et. al [2] is that what convolutional neural networks (CNNs) learn about content can be identified and separated from what they learn about style.

To understand this, we start first with the intuition behind why CNNs are so good in image processing tasks. CNNs consist of layers of units with convolutions that can process local patterns in specific parts of the image [9]. These local patterns comprise features like edges and textures that recur in other parts of the image and combine in deeper layers for more complex structures. Layers are then sets of units, where each unit observes certain patterns and filters out other patterns, the entire layer acting as a transformation on the input image into representations [9].

This is particularly relevant because by preserving the layer and its outputs (or feature maps), we can infer the information contained in each layer by reconstructing the image through the layer. We reconstruct an image by finding another image that has the same values in our layer of choice. This is discussed more formally in the succeeding sections. As we get deeper in the network, the layers tend to care less about the exact pixel values and more about the arrangements of said pixels into patterns and objects - in other words, the content of the image.

3.1 Content Representation

The content representation of an image p is defined as the values in some layer l of a pre-trained CNN, where l is a hyperparameter [2].

Suppose that, after feeding some input image p, layer l has N_l feature maps, each of size $a_l \times b_l$. We can flatten these feature maps such that each feature map is a row of length $M_l = a_l \times b_l$. We can then stack each row onto a matrix F^l of size $N_l \times M_l$. We will refer to this matrix as the content representation of the input image p. Note that F^l_{ij} is the activation of the i^{th} filter at position j in layer l.

Now, to reconstruct the "content" of an image p, we take a white noise image and use gradient descent to make the content representation of the white noise image sufficiently close to the content representation of our content image. In particular, the loss function at layer l between the original image p with content representation P^l and our trained image x with content representation F^l is given by

$$\mathcal{L}_{\text{content}}(p, x) = \frac{1}{2} \sum_{i,j} \left(F^l_{ij} - P^l_{ij}\right)^2 .$$

We refer to this function as the content loss. By choosing different values for the layer l, we obtain varying results. To see this, we use the VGG19 model to reconstruct the content of the image in Fig. 3 from various layers. Results are shown in Fig. 4.

Figure 3: Original content image

(a) (b) (c) (d) (e)

Figure 4: Content reconstructions of an image of a monkey using (a) conv_1_2, (b) conv_2_2, (c) conv_3_2, (d) conv_4_2, and (e) conv_5_2

Note that as we get deeper into the network, the reconstruction becomes less exact, but the overall outline of the monkey remains. Ideally, we would like our output image to have the outline of a monkey without the exact pixel values of the original content image, which we would like to replace with the style (texture, color, and stroke) of another image. Thus, an ideal layer choice for a content representation would be a layer close to "conv_4_2," which happens to be the layer of choice in [2].

3.2 Style Representation

Style representation is less straight forward. The idea for the style representation comes from a previous paper written by the same authors, Gatys et al. [3], where they were able to synthesize textures from an arbitrary image using Gram matrices (see Fig. 5).

(a) Original image (b) Synthesized texture

Figure 5: Synthesized texture of van Gogh's Starry Night

Consider some input image a, layer l, and its subsequent content representation F^l. The Gram matrix G^l is defined as the correlations between the feature maps in layer l. That is, G^l_{ij} is the inner product between the flattened feature map i and j in layer l, or alternatively row i and column j in F^l. More formally,

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}.$$

We refer to the Gram matrix G^l as the style representation of the image at layer l. The correlations between feature maps capture the style of the image, as opposed to the content, focusing on the colors and flourishes that make up the style. This was originally motivated by [5], which suggests that texture is just some summary statistic over a set of pixels. Gatys et. al [2] argue that style

is merely a kind of visual texture, and use Gram matrices to represent it. The intuition for why Gram matrices are used in particular is that they are spatially invariant, focusing on what feature maps light up together. This focuses on common patterns in the image seen at different levels in the network hierarchy, and comprises what we figure to be style.

Then, similar to the "content", we can reconstruct the "style" of an image a by taking a white noise image and using gradient descent to make the style representation of the white noise image sufficiently close to the style representation of our style image. In particular, the loss function at layer l between the original image a with style representation A^l and our trained image x with style representation G^l is given by

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

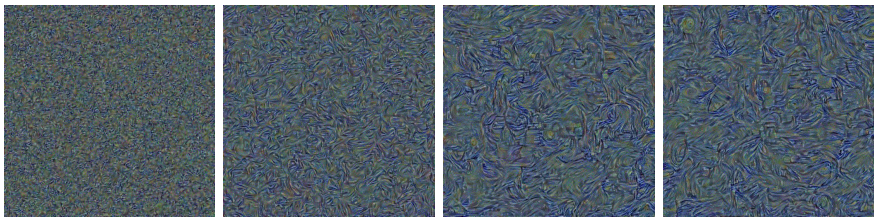
However, since we would like the style representation to consider both global details and textural details at a fine level, including representations from different levels of the hierarchy can provide a smoother visual experience (and probably one that looks nicer). Hence, we would include more layers from 1 to L (where L is the number of layers) to the loss and weight them with weights w_l (as hyperparameters). Thus, the final loss function is given by

$$\mathcal{L}_{\text{style}}(a, x) = \sum_{l=1}^{L} w_l E_l,$$

which we call the style loss. The default in [2] is to weight all layers equally. Similar to the content, the choice of layers also affects the resulting texture. To see this, we use the VGG19 model to reconstruct the style of the image in Fig. 6. Results are shown in Fig. 7.

Figure 6: Original style image

For simplicity, we choose various prefixes of layers conv_1_1, conv_2_1, conv_3_1, conv_4_1, and conv_5_1. We find that including more layers in our style loss results in visually better textures. In fact, the layers of choice in [2] happens to be all of conv_1_1, conv_2_1, conv_3_1, conv_4_1, and conv_5_1.



(a) (b) (c) (d)

Figure 7: Style reconstructions of an image of van Gogh's Starry Night using layer (a) conv_1_1 and conv_2_1, (b) conv_1_1, conv_2_1, and conv_3_1, (c) conv_1_1, conv_2_1, conv_3_1, and conv_4_1, and (d) conv_1_1, conv_2_1, conv_3_1, conv_4_1, and conv_5_1

3.3 Putting It Together

Neural style transfer aims to generate images that simultaneously minimize the distance to both the content representation (content loss) and the style representation (style loss). Thus, given a content image p and a style image a, the loss function to be minimized is

$$\mathcal{L}_{\text{total}}(p, a, x) = \alpha \mathcal{L}_{\text{content}}(p, x) + \beta \mathcal{L}_{\text{style}}(a, x)$$

where α and β are weighting factors for the content and style loss respectively. The values for α and β can be varied depending on the desired emphasis on content or style. Increasing the ratio $\frac{\alpha}{\beta}$ places more emphasis on displaying the content while decreasing the ratio places more emphasis on recreating the style. The original paper [3] compared several ratios in their discussion, specifically 1×10^{-3}, 8×10^{-4}, 5×10^{-3}, and 5×10^{-4}.

4 Methodology

As prefaced in sections 2.1 and 2.2, this paper compares two models, VGG19 and DenseNet201. We specifically elected to use DenseNet201 as the base model to compare to VGG19 due to the availbility of its implementation on PyTorch and the relatively similar structure of its convolutional layers. The specifics of the two models are outlined in Table 1.

For VGG19, the chosen layers for the content and style representations follow from the specifications outlined in the original paper [2]. For the DenseNet201 layers, these were selected after comparing the reconstructed content and synthesized texture of different selections of dense layers / combinations of dense layers and then choosing the ones that we deemed to be most visually promising. Fig. 8 shows the resulting style reconstruction of the DenseNet201 model with the chosen layers.

For both models, the ratio of the content and style loss weights $\frac{\alpha}{\beta}$ is noticably smaller than any of the ratios investigated by [2]. This is because unlike in the

	Content Representation Layer	Style Representation Layers	Loss Function Weights
VGG19	conv_4,2	conv_1_1, conv_2_1, conv_3_1, conv_4,1, conv_5_1	$\alpha = 1$ $\beta = 10\hat{}6$
DenseNet201	dense_4_31	dense_2_12, dense_3_48, dense_4_32	$\alpha = 1$ $\beta = 10\hat{}8$

Table 1: Details for the VGG19 and DenseNet201 models

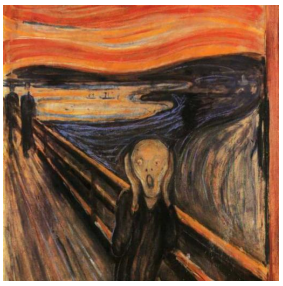
Figure 8: DenseNet201 style reconstruction

original paper, where the final image is generated starting from white noise, our model training process begins with the content image itself. Since the main content is already present and needs to be maintained instead of fully recreated, we opted for a greater emphasis on style over content. We also observed in preliminary testing that the initial style loss values for DenseNet201 were very small to begin with, motivating our choice of a large style loss weight versus VGG19.

In terms of results, we will look at the quantitative and qualitative performance of these models in NST on three style images and one content image, in order to compare the transfer of style. These images can be found in Figure 9. They were resized and center cropped to all be the same size.

(a) Claude Monet's Impression, Sunrise (1872)

(b) Vincent Van Gogh's The Starry Night (1889)



(c) Edvard Munch's The Scream (1893)

(d) Photo of a Monkey from the BBC's Africa: The Greatest Show on Earth (2013)

Figure 9: Style and Content Images for Neural Style Transfer

5 Qualitative Discussion

Looking at the outputs of both the VGG19 and DenseNet201 models across all three styles (See Fig. 12 in Appendix), the two models perform the neural style transfer at an adequate level. In all outputs, the main content of the monkey's face is kept visually coherent, while also being rendered in a style that clearly evokes the painting used, particularly with the coloring. However, looking closely at the finer details of the outputs, we contend that VGG19 generally outperformed DenseNet201.

First, while both models transferred the colors of the style paintings well, VGG19 was better at capturing the texture of the paintings, such as the strokes and brushwork. This is most apparent in the Starry Night outputs, where the output of VGG19 demonstrates the characteristic whorls and swirls of Van Gogh's work across the monkey's fur and facial features. This contrasts with the outputs of DenseNet201, which appears to maintain the "photorealistic" texture of the content photo, making the monkey seem more recolored instead of painted.

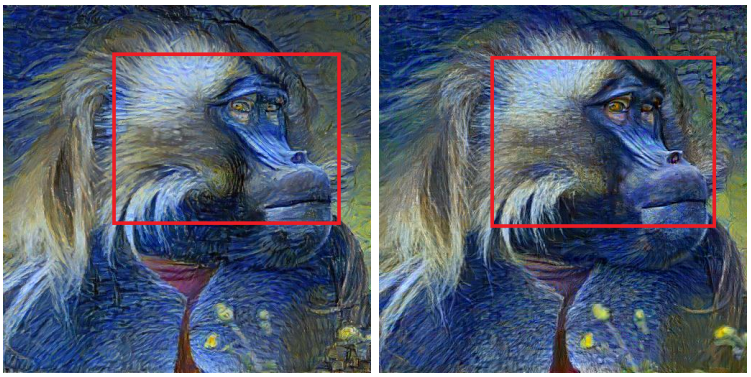


Figure 10: Texture comparison between VGG19 (left) and DenseNet201 (right) for the Starry Night style.

Second, we also feel that VGG19 was better at recreating the background. In the original content image, the background is heavily blurred out and the focus is almost entirely on the monkey. In the VGG19 outputs, the backgrounds are kept relatively light and minimalist, while still adhering to the painting style. In comparison, the backgrounds created by DenseNet201 are more visually noisy, particularly with The Scream and Starry Night. In fact, it seems that much of the texture went to the background instead of the monkey for these DenseNet201 outputs.

Third and last, VGG19 performed much better in rendering the foreground elements outside of the monkey. While not the focus of the photo, we can observe in the bottom-right of the content image that there are some out-of-focus flowers and greenery in front of the monkey. In the DenseNet201 outputs, particularly with The Scream and Starry Night, they appear hardly affected

by the painting style at all, making them stick out and look out of place. In comparison, VGG19 does a better job applying the style and blending in these elements with the rest of the image.

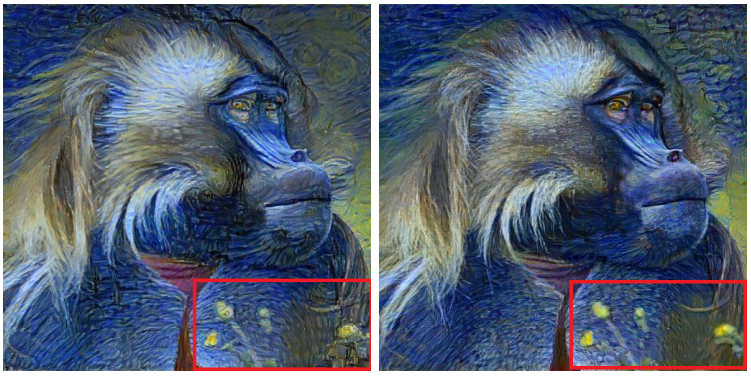


Figure 11: Foreground comparison between VGG19 (left) and DenseNet201 (right) for the Starry Night style.

Given that there is some degree of randomness in the final output of the models, we recreated the images several times using both models. From these runs, we consistently observed the same qualitative differences between VGG19 and DensNet201, and so we maintain the VGG19 is the better performer for style transfer.

6 Quantitative Discussion

In general, it is not easy to find a straightforward explanation for why one model does "better" than another. Due to various factors such as randomness in the training, the selection of hyperparameters (e.g., content layers, style layers, weighting coefficients, etc.), and even the choice of picture, it is hard to generalize and come up with a definitive answer.

At a high level, the main reason why VGG19, despite its smaller size, would perform better than DenseNet201 is due to the separability of content and style layers. VGG19, with its simpler architecture, is likely to produce feature maps that are more localized and less correlated across channels compared to DenseNet201. This is especially true because of the interconnections between all layers, making it harder to separate and manipulate content and style features during the style transfer process. This would be reflected in the clearer distinction between the color, texture, and stroke aspects ported over in the VGG19 outputs, compared to the mild carry-over of some content elements from the style image in the DenseNet201 outputs (like some "stars" from Starry Night, or the clouds from Sunrise).

Wang et. al [8] investigates the difference in effectiveness between VGG19 and ResNet, and comes to a conclusion relevant to our study. ResNet, a neural

network architecture, mainly differs from DenseNet by the connectivity pattern, using residual connections (adding the input to a layer is added to the output of that layer) instead of skip connections (as mentioned, concatenating input from all previous layers into succeeding layers). ResNet was discovered to perform significantly better at style transfer tasks when all residual connections were removed, with the output significantly preferred by human selectors over ResNet outputs with residual connections included.

Why do residual connections significantly worsen performance? The authors identify that performing the addition of inputs after batch normalization and ReLU activation causes residuals to grow from layer to layer, creating a cascading effect of large activations in deeper parts of the network. The authors tracked the evolution of activations through the network, observing a "whack-a-mole" effect where attempts to mitigate peaks in intermediate layers resulted in larger peaks in other channels of subsequent layers. This increased maximum causes single feature channels with large residuals to dominate the activation, resulting in lower entropy and causing the Gram matrix to focus on pairs on locations of strong activations with high correlations. This has the consequence of reducing a lot of information meant to be distilled in the network and failing to capture the long-term correlations crucial for texture and style. According to the authors: "By giving disproportionate emphasis to these location pairs, the optimization overfits on a few style patterns, ignoring most of the remaining."

Our suggestion is that skip connections will have a similar, but less pronounced negative effect on the performance of models in style transfer. This is because while they lack the additive nature of residual connections that creates peaks in Gram matrices, the information from the input can still flow directly to the output without undergoing the non-linear transformations applied in the main path. As a result, if the input and output distributions differ significantly, the addition of the skip connection can lead to large residuals. In addition, skip connections still create interconnections between earlier and later layers, increasing complexity substantially and causing the Gram matrix to potentially have a harder time spotting the long-term correlations useful for style transfer.

7 Conclusion

Both the VGG19 and DenseNet201 models were adequately able to perform NST over a picture of a monkey across styles from three different paintings. However, closer qualitative inspection showed that VGG19 generally performed better than DenseNet201, particularly in terms of the translation of style from the paintings to the content photograph. Further analysis of the differences in architecture between the two models lead us to believe that this discrepancy is likely caused by the skip connections present between the dense layers in DenseNet201, as supported by literature regarding the residual connections in the similarly structured ResNet.

References

[1] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[2] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "Image Style Transfer Using Convolutional Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 2414–2423. ISBN: 9781467388511. DOI: 10.1109/CVPR.2016.265. URL: http://ieeexplore.ieee.org/document/7780634/.

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. *Texture Synthesis Using Convolutional Neural Networks*. 2015. arXiv: 1505.07376 [cs.CV].

[4] Gao Huang et al. "Densely Connected Convolutional Networks". In: (Jan. 2018). DOI: 10.48550/arXiv.1608.06993. URL: http://arxiv.org/abs/1608.06993.

[5] Randi C. Martin and James R. Pomerantz. "Visual discrimination of texture". en. In: *Perception & Psychophysics* 24.5 (Sept. 1978), pp. 420–428. ISSN: 0031-5117, 1532-5962. DOI: 10.3758/BF03199739. URL: http://link.springer.com/10.3758/BF03199739.

[6] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[7] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[8] Pei Wang, Yijun Li, and Nuno Vasconcelos. "Rethinking and improving the robustness of image style transfer". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 124–133. ISBN: 9781665445092. DOI: 10.1109/CVPR46437.2021.00019. URL: https://ieeexplore.ieee.org/document/9577720/ (visited on 12/12/2023).

[9] Rikiya Yamashita et al. "Convolutional neural networks: an overview and application in radiology". en. In: *Insights into Imaging* 9.4 (Aug. 2018), pp. 611–629. ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9. URL: https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9.

8 Appendix

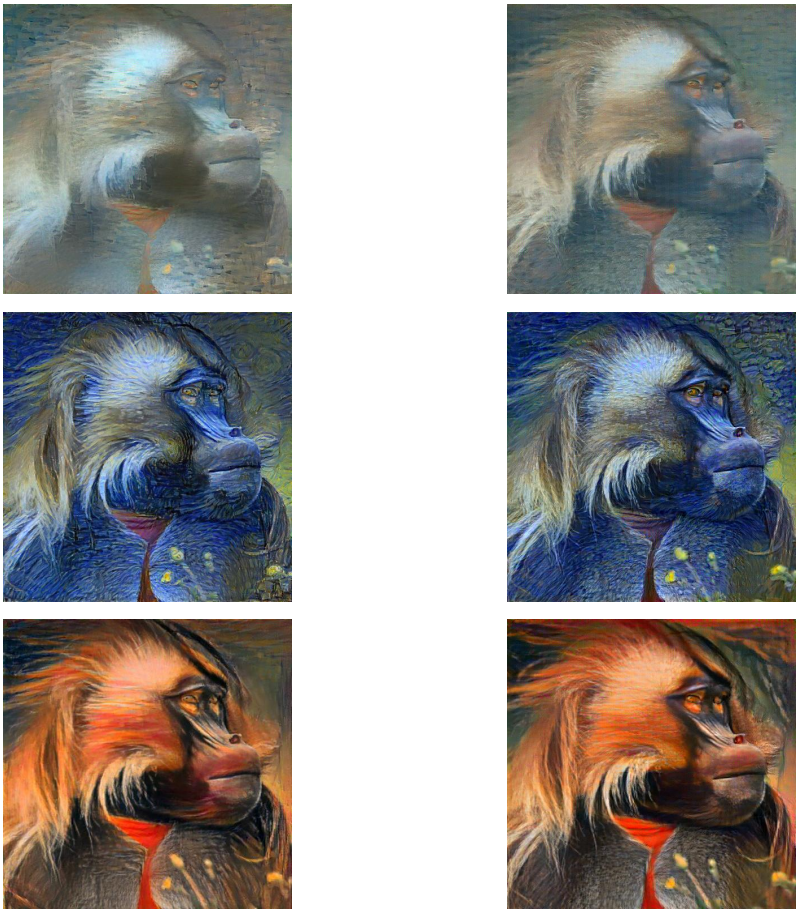


Figure 12: Outputs for VGG19 (left) and DenseNet201 (right) in Neural Style Transfer