Interpretable Neural Style Transfer using VGG Architectures

Siddhant Joshi[1] **William Hu**[1]

Abstract

Rendering the semantic content of an image in various styles poses a unique image processing challenge that has recently garnered significant attention in the field of computer vision (CV). Gatys et al. published one of the first papers concerned with this application of deep learning, dubbed Neural Style Transfer (NST), opening many avenues for further research. In this paper, we will primarily focus exploration into the algorithm's interpretability, providing our own survey of experiments with varying degrees of key parameters involved in a classic VGG-19-based NST algorithm. Our objective is twofold: to gain deeper insights into the algorithmic NST decision-making process and to design a new algorithm that enhances fidelity and content preservation in stylized images. Not only is our proposed NST algorithm able to achieve lower total losses than the baseline, but also produce more visually appealing stylized images with a simple feed-forward neural network.

1. Introduction

Gatys et al. recently approached this task using deep neural networks, showing that they can not only encode the content but also the *style* information of images[7]. Using this concept, they published their cornerstone work on NST utilizing VGG-19 to effectively separate content and style information from images and combine these representations into new, stylized images. Since then, subsequent research focused on improving the efficiency and speed as well as enabling more customization into the stylistic focuses of the algorithms[1][4][5][9].

Before understanding the goal of our study, we must establish an intuition behind the original NST algorithms. As described in [7], NST can be reduced down to an optimization problem involving four components: a content image (I_C), a style image (I_S), an input image (I_i), and a pre-trained feed-forward convolutional neural network (CNN). All three images are fed into the CNN, such as VGG-19, to extract characteristic features. Generally, successive convolutional layers within a network capture increasingly complex features, abstracting pixel-wise information to representations that capture the high-level content of an image. Stylistically, various relevant information are encoded throughout the feature maps of each layer. To effectively capture the dispersed style information, [7] suggest considering the feature maps of every convolutional layer and combining their information into Gram matrices to allow for holistic consideration of lower and higher-level styles. As such, extracting content information from deeper convolutional layers would be useful for content preservation whereas consulting Gram matrices would be useful for extracting style. These "feature summaries" are then compared to those of I_i via a squared-loss calculation, resulting in a content loss ($\mathcal{L}_C$) and a style loss ($\mathcal{L}_S$) that will be used in gradient descent to update the pixels of the I_i to better represent the style and content information. This entire process is repeated over multiple transfer iterations, eventually converging toward a final stylized output image (I_O). Fig. 1 presents a full diagram of the algorithm.

Our study is concerned with the composition of this algorithm, specifically in unveiling the relationships between adjustable parameters and their individual impact on the quality of style transfer. Additionally, we explore error metrics against subjective visual appeal of the output images, seeking potential for a reconciliation between the quantitative and qualitative aspects of this application.

2. Methodology

As in Gatys et al.'s original study, our experiments will be generated with the VGG-19 network, which was trained to perform object recognition on the ImageNet dataset[2]. We will remove the fully-connected (FC) layers and final softmax activation layer from the model, isolating the pre-trained convolutional blocks for the algorithm. Additionally, we convert every MaxPool2d layer into AvgPool2d layers as average pooling was found to produce higher quality outputs in previous studies[7].

[1]Department of Cognitive Science and Halıcıoğlu Data Science Institute, University of California, San Diego, La Jolla.

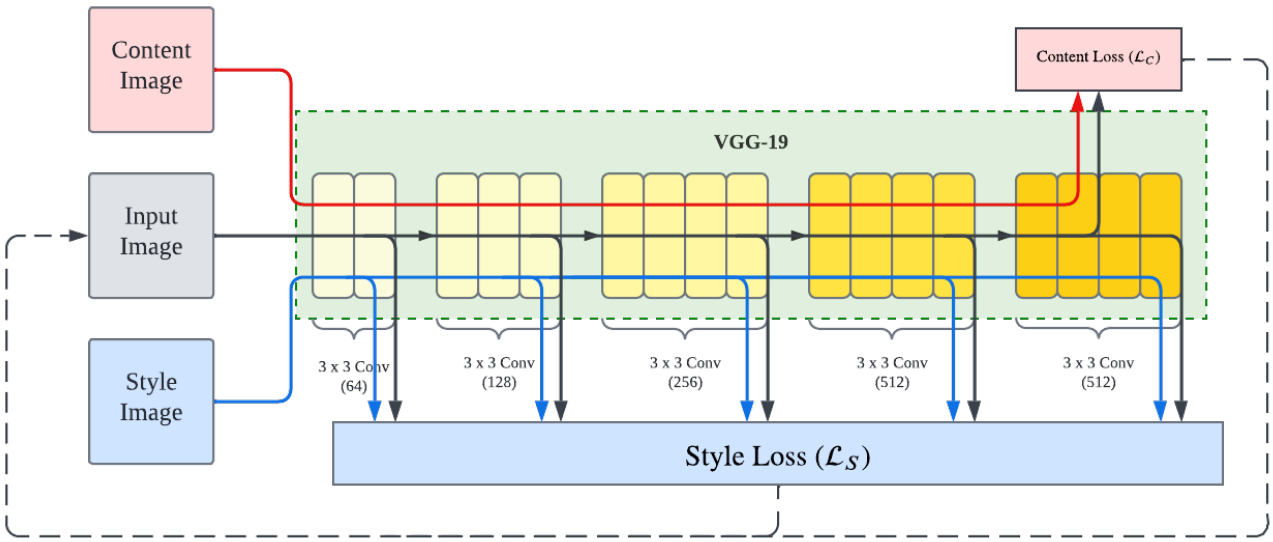


Figure 1. Input, content, and style images are all passed into the VGG-19 network where feature maps from convolutional layers are pulled out and converted into their respective losses. Both $\mathcal{L}_C$ and $\mathcal{L}_S$ are fed into the input image for pixel-wise updating via gradient descent.

The following table describes the parameters of the baseline model for this study.

Parameters	
Number of Iterations	300
Content Error Layer	conv_4
Style Error Layers	conv_1
	conv_2
	conv_3
	conv_4
	conv_5
Content Weight	1
Style Weight	1000000
Loss function	L2
Optimizer	L-BFGS

Table 1. Parameter Settings

To limit computational complexity, we will study the effect of each parameter of interest in isolation of the others, assessing performances across each independent parameter and recording total loss of the stylized image. Furthermore, we will be maintaining the same content and style images throughout our experiments to ensure results are quantitatively and qualitatively comparable (Fig. 2). The following subsections explain the parameters we will be testing and provide rationale for design choices.

2.1. Number of Transfer Iterations

Current literature does not seem to have specific preferences for the number of transfer iterations. As such, we will arbitrarily assess performance on transfer iterations from the range {100, 200, 300, 600, 1000, 2000}.

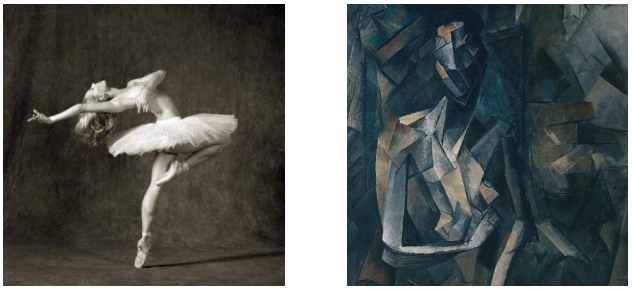


Figure 2. Baseline I_C (left) and I_S (right).

2.2. Error Layers

When it comes to assessing transfer quality, the choice of layer with which to construct Gram matrices is critical. Aside from convolutional layers, we will also be assessing representation quality in ReLU layers within the VGG-19 network. This would be the most computationally intensive aspect of our study, as we will be assessing all 31 combinations of convolutional and ReLU layers between both style and content, resulting in 124 total trials.

2.3. Error Weights

Mathematically, the baseline losses for style are scaled to be comparable to that of the content loss. Since the key behind the ”learning” in NST relies on the magnitudes of

these losses, adjusting their scaling factors may prioritize different degrees of content and style in the stylized image. We assessed styles weights in the range {1000000, 100000, 1000, 100} and content weights from the range {1, 5, 10, 20, 100}.

2.4. Stochastic Focus

We also decided to delve into algorithm generalizability by leveraging some common image sampling methods used in image classification to prevent overfitting. Currently, the baseline algorithm considers the entire style image to update a copy of the content image. We introduce randomness into the algorithm to curb these potential issues via the following methods:

2.4.1. Random Style Shifting

In our current process, each transfer iteration is followed by a comparison of pair-wise similarities between corresponding pixels. As such, each pixel is at risk of overfitting to single-pixel value information it receives. We add variability to this process by randomly cropping and rotating the I_i with Kornia library before processing similarity. In doing this, we aim to have the algorithm fit each output pixel to a neighborhood of style pixels, thereby generalizing style throughout the I_i. Since these modifications are to be slight, we decided to consider transformation dimensions of at least 95% of the size of the output image. More specifically, from the range {0.95, 0.96, 0.97, 0.98, 0.99}.

2.4.2. I_i Initialization

We also believe the choice of the initial I_i input plays a role in the quality of the output image. We will test the following three options for I_i:

- The content image (I_C) [Baseline]
- 512×512 image of Gaussian-random pixel values
- 512×512 image with all pixel values set to 0 (black)

2.5. Style Image Type

All images contain style and content aspects, so gauging the separability of these components seems logical. We can broadly classify major art works into one of landscapes or portraits. As such, we will test the baseline on combinations of images from these classes to get a better understanding of how NST style extraction ignores content and vice versa. We introduce the following 4 images, two landscapes and two portraits, and run the algorithm on every pairwise combination (Fig. 3 & 4).

2.6. Loss Function

Aside from the baseline L2 error, we also assessed the performance of L1 error as it is generally preferred for its robustness to overfitting. In this context, we anticipate it to counter the output image overfitting to either style or content information.

Figure 3. I_C Images

Figure 4. I_S Images

2.7. Optimizer

The baseline algorithm uses a Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) optimizer. We will conduct tests using Adam, Adagrad, and standard SGD optimizers from PyTorch, as they are preferred by research when handling image processing tasks.

3. Experimental Results and Discussion

3.1. Baseline Performance

For reference, we generated a baseline output image that achieved the following metrics.

Figure 5. $\mathcal{L}_C = 2.227, \mathcal{L}_S = 0.148$

3.2. Number of Transfer Iterations

After running the baseline model with the proposed transfer iterations, we yielded the following loss curve (Fig. 6).

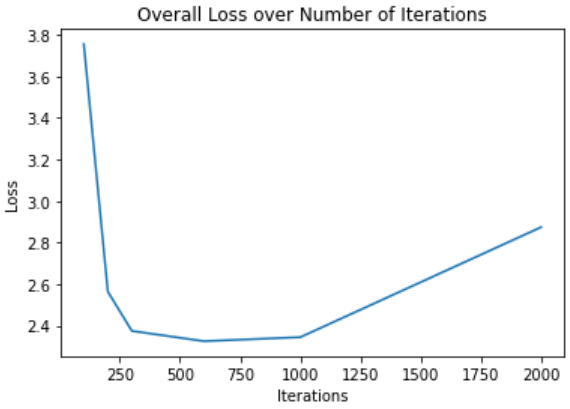


Figure 6.

This loss curve is convex, achieving a minimum around 600 transfer iterations. The parabolic shape seems to be indicative of the algorithm overfitting to the content and style images, achieving convergence at 600 iterations.

3.3. Error Layers

Based on loss metrics alone, the ReLU combinations for both $\mathcal{L}_C$ and $\mathcal{L}_S$ layers consistently outperformed their convolutional counterparts. Visually, extracting content and style information from ReLU layers generated extremely similar results to that of the convolutional layers.

Figure 7. Conv_1 error layer (left) and ReLU_1 error layer (right)

One interesting thing of note however, was that the earlier ReLU layers (ReLU_1 and ReLU_2) achieved the lowest overall content loss of all ReLU trials. In theory, We expected that deeper ReLU layers (ReLU_3 and ReLU_4) would have performed better as their feature maps would be capturing more abstract details from I_C, however that does not seem to be the case, warranting further tuning and discussion.

3.4. Error Weights

Below are some figures of the output with differing style weights (Fig. 8).

Figure 8. Output images of decreasing style weights: 1000000 (top-left), 100000 (top-right), 1000 (bottom-left), 100 (bottom-right)

Below are some figures of the output with differing content weights (Fig. 9).

Figure 9. Output images of decreasing content weights. From left to right: 100, 10, 1.

In both series of experiments, decreasing one weight would increase the prevalence of the features captured by the other weight. For example, decreasing the content weight brought out more of the soft, geometric style of I_S in the output image. With lower relative style weights, the realism of the dancer was prioritized, even resulting in an output that maintained the color scheme of I_C (Fig. 8, bottom-right)

Quantitatively, lower weights in either style or content resulted in decreased overall loss. This makes intuitive sense as the algorithm is focused on optimizing one aspect more than the other, which is an easier problem to solve than attempting to minimize the loss in both content and style simultaneously. However, this behavior highlights the dis-

parity between visual quality and quantitative improvements, specifically that decreasing loss does not necessarily correlate to well-stylized outputs. We will discuss this further in following sections.

3.5. Stochastic Focus

3.5.1. Random Style Shifting

In testing on the range of values for style shifting, we recorded the following losses:

Factor	Total Loss
0.95	36.681
0.96	85.760
0.97	25.836
0.98	131.691
0.99	196.711

Table 2. Loss Across Shifting Factors

From the table, we can see that the losses varied across scaling factors with 97% achieving the lowest.

3.5.2. I_i Initialization

Aside from the baseline initialization of the input image with the content image, we assessed an input of Gaussian-random pixel values and an input of all pixel values set to zero.

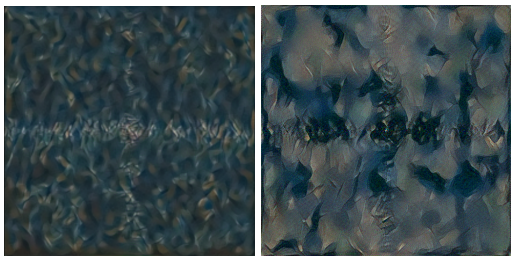


Figure 10. Gaussian-random (left) and zero-pixel initialization (right)

Clearly, these output images preserved little to no content information over the course of 300 transfer iterations. However, there seems to be an interesting blend of the style and input images present in either output. This could hint at a possibility of this particular treatment relying on other algorithmic changes to be more effective in blending style with content. Therefore, we will further explore this as a hyperparameter to assess the viability of our belief in interdependence.

3.6. Loss Functions and Optimizers

We recorded the following results for the different loss functions and optimizers that were tested.

Loss Functions	
L2 (MSE)	2.375
L1 (MAE)	588.243
Optimizers	
L-BFGS	2.375
Adam	33.714
Adagrad	33.184
SGD	37.368

Table 3. Total loss of various loss functions and optimizers

Surprisingly, L1-loss performed significantly worst than L2-loss. Yet, as we saw in Section 3.4, the quality of L1-loss outputs kept/retained/captured more style than that of the L2-loss outputs (Fig. 11) Most notably, the fine details of the figure's hands and face are less prioritized in the L1 output, an arguably good sign of effective stylization.

Figure 11. L1-loss output (left) compared to L2-loss output (right)

Regarding choice of optimizer, L-BFGS outperformed all other optimizers based on loss alone. However, upon inspection of the output images, L-BFGS appears to have a lower quality transfer (Fig. 12).

Figure 12. Output images of the different optimizers: L-BFGS (top-left), Adagrad (top-right), Adam (bottom-left), and SGD (bottom-right)

Similar to our interpretations of L1 and L2-loss outputs, the alternative optimizers appear to smooth out details in I_C and bring focus to the overall structure of the dancer, which is preferred given the abstract style image.

3.7. Style Image Type

When assessing the impact of the contents of style images, we analyzed style quality on four different combinations; transferring the style of a cubist portrait onto another portrait and landscape:

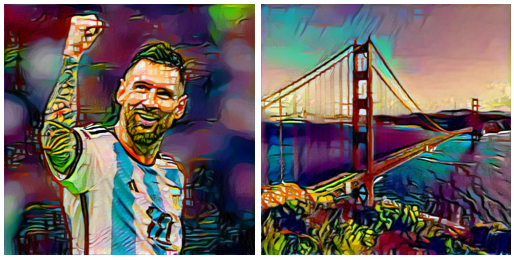


Figure 13. Applying the cubist portrait style onto another portrait (left) and a landscape (right)

And transferring the style of an impressionist landscape onto the same portrait and landscape:

Figure 14. Applying the impressionist landscape style onto a portrait (left) and another landscape (right)

Overall, contrasting image types does not seem to negatively impact the visual appeal of the output images. In Fig. 13, we can see textures and colors from the cubist portrait were represented nicely in Messi's tattoo and the greenery beside the Golden Gate Bridge. Furthermore, the fuzziness of the impressionist landscape was applied effectively on both output images, especially the hints of the pink flowers scattered throughout the horizon of the Golden Gate Bridge.

We acknowledge that these outputs could appear more as an application of a filter over the content image, rather than a blending of style and content. Given that the algorithm is still able to extract and separate style and content representations, we believe the distinction between a style *overlay* and a style *transfer* could be explored as an interaction between parameters.

3.8. Discussion

Over the course of the experimentation we conducted with various parameters of the original baseline algorithm, we discovered some notable independent impacts on output images. Most notably, we observed a trade-off between style and content weights that can be leveraged as a hyperparameter to improve transfer quality. In giving more importance to either aspect, the algorithm could be tuned to specific images based on user-defined requirements. A common theme across multiple experiments was that lower total loss did not necessarily result in the best stylized output. This challenge is unique to the use-case for NST, as subjective evaluations of the output cannot be ignored when the goal is to apply a loosely-defined style to an equally ambiguous content. Additionally, our experimental design and observations beg the consideration of parameter interactions. Our experimentation assessed the parameters in isolation, preventing us from taking into account any further positive (or negative) implications from considering them simultaneously. We saw most promise of this concept when considering I_C-I_S contrast and different I_i initializations. It is well-established in machine learning theory that random, unbiased input leads to more reliable results, so to witness the exact opposite in an isolated setting should not be enough to rule out its effectiveness when incorporated with other parameters.

4. Revising the Algorithm

After experimenting with multiple parameters and analyzing the relationships between them, we propose an improved NST algorithm based on some key findings. Our final algorithm will include a random style shifting before each transfer iteration. Since Adam, Adagrad, and SGD all generated similar results, we opted for an Adam optimizer as it is common practice in CV research. Content weights, style weights, and I_i initialization will be treated as hyperparameters and will be tuned with every combination of

style and content image. We opted for feature extractions via the ReLU layers, because despite producing visually similar results as convolutional layers, the nature of the operation would make running the algorithm less computationally intensive. Table 4 describes the fixed parameters of our proposed revision.

Parameters	
Number of Iterations	600
Content Error Layer	ReLU_5
Style Error Layers	ReLU_1
	ReLU_2
	ReLU_3
	ReLU_4
	ReLU_5
Optimizer	Adam (lr: 0.05)
Loss Function	L1

Table 4. Revised Parameter Settings

4.1. Results

After feeding and tuning our algorithm for various combinations of content and style images, we achieved the following output images (Fig. 19). We also report $\mathcal{L}_C$ and $\mathcal{L}_S$ for the baseline and revised algorithms for a quantitative comparison (Table 5).

Row	Baseline $\mathcal{L}_S$	Baseline $\mathcal{L}_C$	Our $\mathcal{L}_S$	Our $\mathcal{L}_C$
1	0.265	2.352	3.276	0.916
2	4.370	13.113	1.575	0.933
3	5.811	19.208	4.115	3.816
4	4.707	13.526	2.861	7.440
5	4.560	20.193	10.701	0.879

Table 5. Loss Comparison Across Results

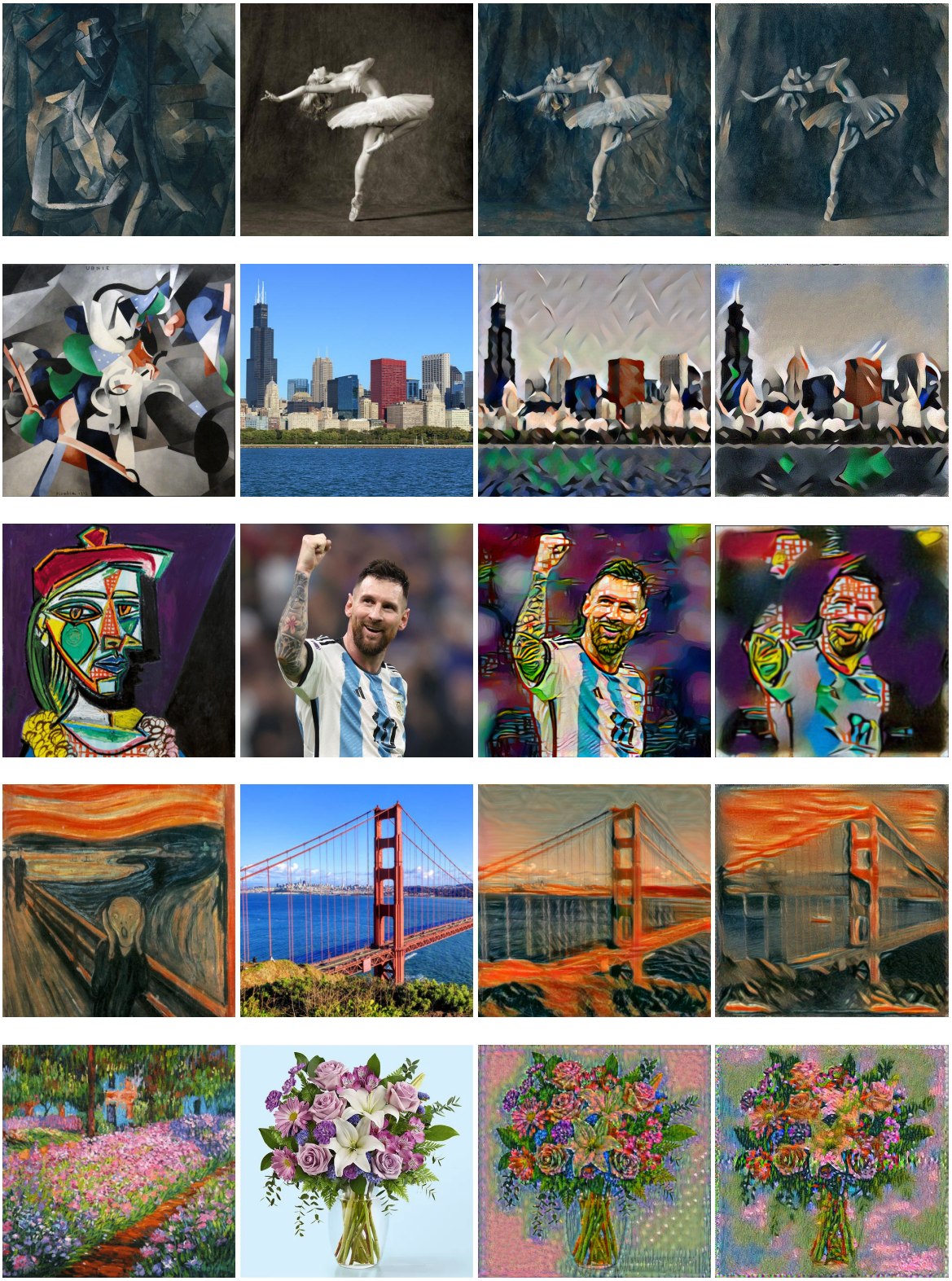
4.2. Discussion

Overall, our revisions significantly reduced total loss when compared to the baseline and produced more visually appealing results. Revisiting our decision to continue investigating different I_i initializations proved beneficial in enhancing transfer quality. For instance, applying the baseline style image onto the dancer with a zero-pixel input image generated a depiction of the dancer built from the irregular polygons present in the style image, with blue streaks and orange highlights. We theorize this was likely due to the algorithm having to reconstruct *both* style and content onto an uninformative input image, encouraging it to better learn the style and incorporate the patterns effectively into pieces of content.

5. Conclusion

In this paper, we conducted a rigorous assessment of the individual impacts of common parameters on an original NST algorithm proposed by Gatys et al. and devised an improved algorithm from our findings[7]. We discovered a trade-off between style and content weightages, efficient feature extraction via ReLU layers, and influence of input initialization.

In future research directions, we hope to consider experimental designs that take parameter interactions into account. This way, we may glean a better understanding of any potential interplay that may have been missed by our computational limitations. Our study only tested 512×512 images and transformed any nonconforming inputs to these dimensions. Exploring algorithmic adjustments to accommodate photos of higher fidelity may lead to different outputs.



. Style . Content . Gatys et al. . Ours

Figure 19. Results Array

Software and References

Software

All of the code for this report was run in Jupyter notebooks on UCSD's Datahub. The built-in VGG-19 network from PyTorch was used for the NST algorithm implementation. We also utilized Kornia, an open source Python package specialized in computer vision.

References

[1] A. J. Champandard, "Semantic style transfer and turning two-bit doodles into fine artworks," (2016) https://arxiv.org/abs/1603.01768

[2] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "ImageNet classification with deep convolutional neural networks," (2017) https://doi.org/10.1145/3065386

[3] C. Olah, A. Mordvintsev, and L. Schubert, "Feature Visualization", (2017) https://distill.pub/2017/feature-visualization/

[4] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. "State of the "Art": A Taxonomy of Artistic Stylization Techniques for Images and Video", (2013) doi: 10.1109/TVCG.2012.160

[5] J. Johnson, A. Alahi, L. Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," (2016) https://arxiv.org/abs/1603.08155.

[6] K. Simonyan, A. Vedaldi, and A. Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", (2013), arXiv:1312.6034

[7] L. A. Gatys, A. S. Ecker and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," (2015) doi: 10.1109/CVPR.2016.265.

[8] V. Dumoulin, J. Shlens, and M. Kudlur. "A Learned Representation for Artistic Style", (2017) https://arxiv.org/abs/1610.07629

[9] X. Huang, S. Belongie, "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization", (2017) https://arxiv.org/abs/1703.06868