Fall 11-21-2016

An Exploration of Style Transfer Using Deep Neural Networks

Cameron Y. Smith

Follow this and additional works at: http://digitalrepository.unm.edu/cs_etds

Recommended Citation

This Thesis is brought to you for free and open access by the Engineering ETDs at UNM Digital Repository. It has been accepted for inclusion in Computer Science ETDs by an authorized administrator of UNM Digital Repository. For more information, please contact disc@unm.edu.
Cameron Y. Smith

Computer Science

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Trilce Estrada, Chairperson

Thomas Caudell

Yin Yang
An Exploration of Style Transfer Using Deep Neural Networks

BY
Cameron Smith
Bachelor of Science, University of New Mexico, 2014

THESIS
Submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the subject of
Computer Science

University of New Mexico
Albuquerque, New Mexico
December 2016
I would like to thank my thesis adviser Professor Estrada of the Department of Computer Science at the University of New Mexico. I would also like to acknowledge my committee members: Professor Yang and Professor Caudell of the Department of Electrical and Computer Engineering at the University of New Mexico. I am gratefully indebted to their valuable comments on this thesis.

I would also like to thank the members of the research group for their helpful feedback: Jeremy, Jacob, Matt, Martha, Haleh, Hooman, Xinyu, and Andres.

Finally, I must express my gratitude to Deirdre, Sheldon, Vishwanath, Kartik, Noah, and my parents for their support and encouragement.
An Exploration of Style Transfer Using Deep Neural Networks

BY

CAMERON SMITH

B.S., COMPUTER SCIENCE, UNIVERSITY OF NEW MEXICO, 2014
M.S., COMPUTER SCIENCE, UNIVERSITY OF NEW MEXICO, 2016

ABSTRACT

Convolutional Neural Networks and Graphics Processing Units have been at the core of a paradigm shift in computer vision research that some researchers have called “the algorithmic perception revolution.” This thesis presents the implementation and analysis of several techniques for performing artistic style transfer using a Convolutional Neural Network architecture trained for large-scale image recognition tasks. We present an implementation of an existing algorithm for artistic style transfer in images and video. The neural algorithm separates and recombines the style and content of arbitrary images. Additionally, we present an extension of the algorithm to perform weighted artistic style transfer.
## Contents

1 Introduction 1

2 Background 3

2.1 Neural Networks for Computer Vision 4

2.1.1 Neural Networks 4

2.1.2 Deep Neural Networks and Deep Learning 6

2.1.3 Stochastic Gradient Descent 7

2.1.4 Biological Basis 8

2.1.5 Convolutional Neural Networks for Computer Vision 8

2.1.6 Network Models 10

3 Theory and Design 11

3.1 Style Transfer in Images 12

3.1.1 Content Representation 13

3.1.2 Style Representation 14
5.1 Conclusion ................................................................. 47

APPENDIX A APPENDICES .............................................. 49

A.1 Hardware and Software Specifications ......................... 49
A.2 Style Images ............................................................ 51

REFERENCES .................................................................. 54
Listing of figures

2.1 Feedforward Neural Network ........................................... 4
2.2 Artificial Neuron ........................................................... 5
2.3 Convolutional Layer ....................................................... 9

3.1 Style and Content Representations ..................................... 16
3.2 Optical Flow and Temporal Consistency ............................... 20
3.3 Preservation of Original Colors ......................................... 21
3.4 Weighted Style Transfer .................................................. 23

4.1 Qualitative Comparisons of Implementations ......................... 28
4.2 L-BFGS and Adam Optimizer Convergence .......................... 30
4.3 Individual Loss Convergence ............................................ 30
4.4 Relative Weighting and Layer Representations ...................... 32
4.5 Convergence for Gradient Descent Initialization .................. 34
4.6 Initialization of the Gradient Descent ................................ 35
Every age has done its work, produced its art with the best tools or contrivances it knew, the tools most successful in saving the most precious thing in the world — human effort.

- Frank Lloyd Wright, 1901

1

Introduction

This thesis presents an implementation and analysis of a texture synthesis algorithm known as the Neural Algorithm for Artistic Style. The algorithm was introduced by Gatys et al. [1] in September 2015. The algorithm uses state-of-the-art Convolutional Neural Networks to separate and recombine the style and content from arbitrary images to synthesize a new pastiche, an artistic work in a style that
imitates that of another artistic work. Additionally, several extensions of the *Neural Algorithm for Artistic Style* are presented. The thesis is structured as follows:

**CHAPTER 1** provides a brief discussion of the fundamental concepts of artificial neural networks and Convolutional Neural Networks for computer vision and image recognition.

**CHAPTER 2** provides a discussion of the theory and methods behind the following artistic style transfer techniques: Single Style Transfer, Multiple Style Transfer, Video Style Transfer, Preservation of the Original Colors, and Weighted Style Transfer. Single Style transfer is the transfer of artistic style using one style image and one content image. Multiple Style Transfer is the transfer of artistic style using more than one style image. Video Style Transfer is the transfer of artistic style to a sequence of video frames. Preservation of the Original Colors is the transfer of artistic style while preserving the colors of the content image. Weighted Style Transfer is the transfer of artistic style from one or more style images to specified regions in the content image.

**CHAPTER 3** provides the qualitative and quantitative results of the techniques presented in Chapter 2. Additionally, the details of our implementation are described.

The majority of our work consists of reproducing the results presented in [2], [3], [4], and [5] using a different machine learning framework. However, we present two extensions of previous techniques. We present an extension of the content image color preservation technique presented by Gatys et al. [3]. We also present an extension of the weighted style transfer technique presented by Chan et al. [5].
We may be making progress in being able to do things like recognize a cat in a photograph. But there’s a huge gulf between that and doing something creative.

Ken Goldberg, 2015

To understand how Deep Neural Networks can be used to transfer the artistic style of images, it is essential to understand the fundamentals of modern neural networks. CHAPTER 2 provides a brief introduction to some of the fundamental concepts of deep learning and Convolutional Neural Networks for image recognition. For further information, we refer the reader to the textbook [6].
2.1 **Neural Networks for Computer Vision**

2.1.1 **Neural Networks**

Artificial neural networks are computational systems composed of a number of simple, interconnected processing elements, which process information through dynamic state responses to external inputs. Feedforward neural networks or multi-layer perceptrons are models whose goal is approximate a function $f^*$. In the case of a classifier, $y = f^*(x)$ maps an input image $x$ to a class label $y$ [6]. Feedforward networks define a mapping $y = f^*(x; \theta)$ where the parameters $\theta$ are learned to find the best function approximation. These learned parameters are called weights. Feedforward networks are composed of many individual functions organized in a chain of layers. The final layer of a network is called an output layer. In supervised learning, the training data $f(x)$ is used to match $f^*(x)$ [6]. The layers between the input layer and the output layer are called hidden layers. The connections between the hidden layers are learned by a learning algorithm.

![Feedforward Neural Network](image)

**Figure 2.1:** Feedforward Neural Network
Neural networks are inspired by biological neurons. Each layer consists of many processing units which act in parallel as vector-to-scalar functions. These units are based on biological neurons because they receive many inputs and compute an activation value. An artificial neuron is a computation unit whose output is a weighted sum of its input. The input of the activation function $f$ is the weighted sum of each of the inputs $x_i$. The function $f$ is defined as:

$$y = f \left( b + \sum_i w_i x_i \right)$$  \hspace{1cm} (2.1)

![Artificial Neuron Diagram](image)

**Figure 2.2:** Artificial Neuron

The activation functions used in a neural network are determined by the designer. Commonly used functions to model a neuron’s output $f$ as a function of
its input $x$ are:

\[ f(x) = \frac{1}{1 + e^{-x}} \quad \text{(Sigmoid)} \]
\[ f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad \text{(Hyperbolic tangent)} \]
\[ f(x) = \max(x, 0) \quad \text{(Rectified Linear)} \]
\[ f(x) = \min(a, \max(x, 0)) \quad \text{(Bounded Rectified Linear)} \]
\[ f(x) = \log(1 + e^x) \quad \text{(Soft Rectified Linear)} \]
\[ f(x) = |x| \quad \text{(Absolute Value)} \]

2.1.2 **Deep Neural Networks and Deep Learning**

The term “deep learning” is essentially a brand name encompassing techniques using Deep Neural Networks (DNNs) composed of multiple processing layers to learn representations of data with multiple abstraction levels. Deep learning has become a popular machine learning research area due to many recent breakthroughs in image and speech recognition. These breakthroughs have improved the state-of-the-art in problems which stood as formidable obstacles to the machine learning community for decades. Deep learning systems have also outperformed other machine learning techniques in areas like predicting the activity of potential drug molecules, analyzing particle accelerator data, reconstructing brain circuits, and predicting the effects of mutations in DNA [7].

Deep Neural Networks (DNNs) are artificial neural networks with multiple hidden layers of units between the input and output layers. DNNs are trained by
repeatedly inputting training data until the average of an objective function stops decreasing (reaches a minima). In the case of image recognition, images are input into the neural network and the neural network outputs a vector of scores, one for each category. An objective function is computed which measures the error (distance) between the output scores and the desired pattern of scores. Based on the objective function, the neural network adjusts the internal adjustable parameters called weights. The weights are adjusted by computing a gradient vector. For each weight, the gradient vector indicates the amount that the error would increase or decrease if the weight were adjusted by a small amount. The weights are then adjusted in the opposite direction to the gradient vector. When averaged over all training examples, the objective function can be viewed as a hilly landscape in a high-dimensional space of weight values. The negative gradient vector indicates the direction of the steepest slope in the landscape, which takes it to a minima where the output error is low [7].

2.1.3 Stochastic Gradient Descent

The training of Deep Neural Networks is typically done with a procedure called stochastic gradient descent (SGD). SGD consists of showing the input vector some examples, computing the outputs and the errors, computing the average gradient for the examples, and adjusting the weights. The SGD process is repeated for small subsets of the examples until the objective function reaches a local minima. A problem with SGD is the possibility of getting stuck at a poor local minima when a global minima has a lower error. In practice, poor local minima are
rarely a problem in a large network because the network nearly always reaches similar minima. The testing of Deep Neural Networks is done with a different set of examples called a test set, which serves to test if the neural network is making accurate predictions. The examples in the test set are not in the training set.

### 2.1.4 Biological Basis

Artificial neural network research is principally inspired and influenced by neuroscientific research of biological neural networks. Due to a lack of scientific understanding of the intricacies of the brain, the current influence of biological neural networks on DNNs can be analogized to the crude influence of birds on the Wright Brothers first flight at Kitty Hawk, North Carolina in 1903. For example, the convolutional and pooling layers in CNNs are directly inspired by the notions of simple cells and complex cells in visual neuroscience, but the human brains LGN-V1-V2-V4-IT hierarchy in the visual cortex ventral pathway does not have an end-to-end supervised learning algorithm like back-propagation [7].

### 2.1.5 Convolutional Neural Networks for Computer Vision

Convolutional Neural Networks (CNNs) are specialized feed-forward neural networks generally used to process data with a grid-like topology, such as 2-dimensional and 3-dimensional image data [6]. A neural network is considered a Convolutional Neural Network if at least one of the layers uses a specialized linear operation called convolution instead of matrix multiplication [6]. These networks have played an important role in the history of deep learning because
they are considered a rare success of biologically inspired computing and have become widely used in large-scale commercial applications.

Convolutional Neural Network are biologically inspired networks designed to simulate the visual cortex. Convolutional Neural Network architectures consist of a stack of distinct layers that transform the input volume into an output volume through a differentiable function. Local connectivity, 3D volumes, and shared weights are characteristics of Convolutional Neural Networks.

A network is called a Convolutional Neural Network if at least one of the layers is a convolutional layer. The term convolutional layer often describes a set of individual convolution, activation, pooling, and fully-connected layers. Each convolution layer is followed by an activation layer. A pair of convolution and activation layers may be followed by more convolution and activation layers or may be followed by a pooling layer. Figure 2.3 shows the interactions between the convolution, activation, and pooling layers in a typical convolutional layer.
2.1.6 Network Models

There are several well-known Deep Neural Networks trained on large data sets for state-of-the-art image recognition. *LeNet* is perhaps the best known Convolutional Neural Network for classifying the MNIST dataset of handwritten digits. The network was developed in the 1980’s by Yann LeCun. *AlexNet* won the ILSVRC challenge in 2012 by reducing the top-5 error rate to 15.3%. This was a historic event because it popularized the use of Convolutional Neural Networks in image recognition. *ZF Net* won the ILSVRC challenge in 2013. The network was an improvement on Alex by altering the parameters. *GoogLeNet* was the winner of the ILSVRC challenge in 2014. The network substantially reduced the number of parameters.

The Convolutional Neural Network used in the experiments in Chapter 4 is the *VGG-19* network created by the Visual Geometry Group at the University of Oxford [8]. The VGG-19 network consists of 19 weighted layers and over 140 million parameters [8]. The VGG network was a runner-up of the ILSVRC challenge in 2014.
Art is the imposing of a pattern on experience, and our aesthetic enjoyment is recognition of the pattern.

- Alfred North Whitehead, 1954

CHAPTER 3 describes the theoretical concepts underlying the image synthesis algorithm *A Neural Algorithm for Artistic Style* which was first introduced by Gatys et al. in September 2015 [1]. The algorithm uses Convolutional Neural Networks to separate and recombine the style of one image and the content of another im-
age to synthesize a *pastiche*, an artistic work in a style that imitates that of another artistic work. Additionally, the theoretical concepts behind several extensions of the algorithm are presented. These extensions include applying the algorithm to video frames, preserving the colors of a content image during style transfer, and weighting the style transfer according to semantic information.

3.1 **Style Transfer in Images**

When an image $x$ is input into a Convolutional Neural Network, the image is encoded in each layer by the filter responses to the image. A layer $l$ has $N_l$ filters and $N_l$ feature maps of size $M_l$ where $M_l$ is the area $H_l \times W_l$ of the feature map. The activations in a layer $l$ can be vectorized and stored in a matrix $F^l \in \mathcal{R}^{N_l \times M_l}$ where $F_{i,j}^l$ is the activation of the $i$-th filter at position $j$ in layer $l$ [1]. This allows an image representation in a layer $l$ to be encoded and stored in a matrix.

Gatys et al. introduced an algorithm to synthesize a new image by using feature representations of the content of an image $p$ and the style of an image $a$ [1]. We will often refer to $p$ as the “content image”, $a$ as the “style image”, and the generated image $x$ as the “stylized image.” Unlike previous style transfer techniques for non-photorealistic rendering, the style transfer problem was formulated as an energy minimization problem consisting of a content loss and a style loss [1]. The key insights behind the algorithm are:

- The features from the convolutional layers in a Convolutional Neural Network carry information about the *content* of an input image $x$. 

12
• The correlations between the features from the convolutional layers in a Convolutional Neural Network carry information about the style of an input image $x$.

### 3.1.1 CONTENT REPRESENTATION

The content of an image is generated by performing gradient descent from a Gaussian noise image to find an image matching the feature responses to the original image. Let $p$ be the content image and $P^l$ be the feature representation in a layer $l$. Let $x$ be the generated image and $F^l$ be the image’s respective feature representation in layer $l$. The squared-error loss between the two feature representations is:

$$L_{\text{content}}(p, x, l) = K \sum_{i,j} (F^l_{i,j} - P^l_{i,j})^2$$

where $K = \frac{1}{2}$ or $\frac{1}{N_l M_l}$ or $\frac{1}{\sqrt{N_l \sqrt{M_l}}}$. The derivative of the content loss with respect to the layer $l$ is:

$$\frac{\partial L_{\text{content}}}{\partial F^l_{i,j}} = \begin{cases} (F^l - P^l)_{i,j} & \text{if } F^l_{i,j} > 0 \\ 0 & \text{if } F^l_{i,j} < 0 \end{cases}$$

Gatys et al. [1] extracted the content image features from a single convolutional layer. However, the content loss function can be easily extended to include feature representations from multiple weighted-layers using:

$$L_{\text{content}}(p, x, L) = \sum_{l \in L} w_l K \sum_{i,j} (F^l_{i,j} - P^l_{i,j})^2$$

13
3.1.2 **Style Representation**

The style representation is the correlation between the different filter responses in the layers of a Convolutional Neural Network. The feature correlations are given by the Gram matrix $G^l \in \mathbb{R}^{N_l \times M_l}$ where $M_l$ is area $H_l \times W_l$ the feature maps in the layer $l$ and $N_l$ is the number of features maps in the layer $l$ [1]; that is, $N_l$ is the depth of the layer $l$. The Gram matrix entries $G^l_{i,j}$ are the inner product between the vectorized feature map $i$ and $j$ in layer $l$:

$$G^l_{i,j} = \sum_k F^l_{i,k} F^l_{j,k}$$ (3.4)

Let $a$ and $x$ be the original image and the generated image. Let $A^l$ and $G^l$ be the images respective feature representation in layer $l$. The contribution of that layer to the total loss is:

$$E_l = \frac{1}{4N^2_l M^2_l} \sum_{i,j} (G^l_{i,j} - A^l_{i,j})^2$$ (3.5)

and the total loss is:

$$\mathcal{L}_{\text{style}}(a, x, L) = \sum_{l \in L} w_l E_l$$ (3.6)

The derivative of $E_l$ with respect to the layer $l$ is:

$$\frac{\partial E_l}{\partial F^l_{i,j}} = \begin{cases} \frac{1}{4N^2_l M^2_l} ((F^l)^T (G^l - A^l))_{j,i} & \text{if } F^l_{i,j} > 0 \\ 0 & \text{if } F^l_{i,j} < 0 \end{cases}$$ (3.7)
3.1.3  Style and Content Combination

An image is synthesized from a style image $a$ and a content image $p$ by matching the content representation of $p$ and the style representation of $a$. For example, the style image might be a famous artwork and the content image might be a photograph. Framed as an energy minimization problem, a new image $x$ is synthesized by jointly minimizing the distance of the style and content feature representations from Gaussian noise. The combined style and content loss is a linear combination of the style and content losses is:

$$L_{total}(p, a, x, L^C, L^S) = \alpha L_{content}(p, x, L^C) + \beta L_{style}(a, x, L^S)$$

(3.8)

where $\alpha$ is the weighting of the content reconstruction, $\beta$ is the weighting of the style reconstruction, $L^C$ is the set of layers used in the content representation, and $L^S$ is the set of layers used in the style representation [1].

---

**Algorithm 1** Neural Algorithm for Artistic Style

**procedure** NEURAL-STYLE($p, a, x, L^C, L^S, \alpha, \beta$)

$$L_{style}(a, x) \leftarrow \sum_{l \in L^S} w_l \frac{1}{4N_l^2M_l^2} \sum_{i,j} (G_{i,j}^l - A_{i,j}^l)^2$$

$$L_{content}(p, x) \leftarrow \sum_{l \in L^C} w_l \frac{1}{2} \sum_{i,j} (F_{i,j}^l - D_{i,j}^l)^2$$

$$L_{total}(p, a, x, L^C, L^S) \leftarrow \alpha L_{content}(p, x, L^C) + \beta L_{style}(a, x, L^S)$$

**while** convergence criterion not met **do**

$$x \leftarrow \text{MINIMIZE}(L_{total})$$

**return** $x$
Figure 3.1: Style and Content Representations

3.1.4 Noise Removal

The stylized image can contain low-level noise. The appearance and detail of the noise suggests it is caused by the filters in the Convolutional Neural Network. Total variation denoising is an image processing technique for noise removal [9]. For a 2-dimensional image signal, we can introduce a total variation loss function:

\[ \mathcal{L}_{tv}(x) = \lambda \sum_{i,j} (x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2 \]  

where \( x \) is the stylized image and \( \lambda \) is the weight of the total variation loss.

3.2 Style Transfer in Videos

Animations can be created by transferring the style of an artistic image to each frame of an image sequence. This is a challenging problem since the solution of
style transfer is not stable. This section presents two naive approaches and an improved approach which uses a temporal constraint and optical flow estimations.

3.2.1 Naive Approaches

The naive approach for creating an animation is to transfer the style of the artistic image to each frame independently. In this approach the network is initialized with Gaussian noise, the current content frame, the style image, or an interpolation between these. This approach is unsatisfactory because it will result in a video with significant perceptual discontinuities between contiguous frames, creating a flickering effect. These discontinuities are disorienting and aesthetically unpleasant.

Siegel et al. proposed a simple approach to produce a video with more sensible continuities between frames [10]. In this approach the training run for each frame is initialized with the stylized output frame of the previous training run. Instead of initializing the network with each sequential frame, this technique significantly reduces the perceptual discontinuities between contiguous frames which results in a more aesthetically pleasing video. While both of these approaches successfully transfer artistic style to video, the video often contains flickering to the remaining discontinuities between frames.

3.2.2 Optical Flow and Temporal Consistency

Ruder et al. proposed a more advanced technique which takes into account the optical flow and temporal consistency between frames [4]. In this approach frames
are generated by initializing the Convolutional Neural Network with the previous stylized frame $i$ warped to the current frame $i+1$ using the estimated optical flow between the pair of images. The first frame must be initialized from Gaussian noise, the content image, the style image, or an interpolation between these. Initializing the gradient descent with the previously stylized frame warped to the current frame yields some temporal consistency between frames.

To enforce stronger consistency between frames, temporal loss functions are used to penalize regions in the image where the optical flow is consistent and estimated with high confidence. Penalizing points excluding disoccluded regions and motion boundaries allows those to generated while preserving the remaining parts of the image. Ruder et al. define inequalities for marking disocclusions and motion boundaries [4]. The temporal loss function is [4]:

$$L_{\text{temporal}}(x, \omega, c) = \frac{1}{D} \sum_{k=1}^{D} c_k (x_k - \omega_k)^2$$  \hspace{1cm} (3.10)

where $D = W \times H \times C$ is the volume of the image. The weights between the previous image and the current image are set as:

$$c_k^{(i-1,i)} = \begin{cases} 
0 & \text{if } c_k \text{ is a disoccluded region or motion boundary} \\
1 & \text{otherwise}
\end{cases}$$

This weighting penalizes regions where the optical flow is consistent [4]. Combining this temporal loss function with the style loss and content loss gives the
total loss:

\[
\mathcal{L}_{\text{shortterm}}(p^{(i)}, a, x^{(i)}, L^C, L^S) = \alpha \mathcal{L}_{\text{content}}(p, x^{(i)}, L^C) + \beta \mathcal{L}_{\text{style}}(a, x^{(i)}, L^S) \\
+ \gamma \mathcal{L}_{\text{temporal}}(x^{(i)}, \omega^{i}_{i-1}(x^{(i)}), c^{(i-1,i)})
\]  

(3.11)

where \( \omega^{i}_{i-1} \) is the function that warps the previous frame \( i - 1 \) to the current frame \( i \) using the optical flow. Additionally, Ruder et al. presented a long-term loss temporal function that penalizes deviations in the previous frame and also more distant frames [4].

\[
\mathcal{L}_{\text{longterm}}(p^{(i)}, a, x^{(i)}, L^C, L^S) = \alpha \mathcal{L}_{\text{content}}(p, x^{(i)}, L^C) + \beta \mathcal{L}_{\text{style}}(a, x^{(i)}, L^S) \\
+ \gamma \sum_{j \in f: i - j \geq 1} \mathcal{L}_{\text{temporal}}(x^{(i)}, \omega^{j}_{i-j}(x^{(i)}), c^{(i-j,i)}_{\text{long}})
\]  

(3.12)

In the long-term loss function the weights are set as:

\[
c^{(i-j,i)}_{\text{long}} = \max(c^{(i-j,i)} - \sum_{k \in f: i - k > i - j} c^{(i-k,i)}, 0)
\]  

(3.13)

where the \text{max} is taken element-wise.
3.3 Extensions of Neural Style Transfer

Following the landmark paper by Gatys et al. several extensions of the algorithm have been created. This section describes techniques for preserving the colors of the original content image, transferring the styles of multiple style images, and weighting the style transfer to apply the style to specific regions in the image.

3.3.1 Preservation of Color in Style Transfer

The original style transfer algorithm transfers the colors of the style image. For certain style and content image pairs, this can result in an unrealistic or undesirable mapping of the color (Fig. 2.2 b). Gatys et al. [1] presented an extension allowing the colors of the original content image to be preserved in the stylized
image. The technique can be used to transfer the colors either before or after the style transfer synthesis.

Let $x_{rgb}$ be the stylized image and $p_{rgb}$ be the content image. Both images are converted to the Y’UV color space. The luminance and color channels of both images are split. An image $x’_{yuv}$ is created by merging the luminance channel from the stylized image and the color channels from the content image. Finally, the result image $x’_{rgb}$ is created by converting $x’_{yuv}$ to the RGB color space. Gatys et al. used the YIQ color space but other color spaces permitting the separation of the luminance and the color channels like the Y’UV, YCrCb, CIE L*u*v*, and CIE L*a*b* color spaces may also work.

![Content image](image1)
![Style image](image2)
![Stylized image with style image colors](image3)
![Stylized image with content image colors](image4)

**Figure 3.3:** Preservation of Original Colors
Algorithm 2 Convert stylized image to original colors

procedure CONVERT-TO-ORIGINAL-COLORS($x_{rgb}, p_{rgb}$)
\[ x_{yuv} \leftarrow \text{RGB-TO-YUV}(x_{rgb}) \]
\[ p_{yuv} \leftarrow \text{RGB-TO-YUV}(p_{rgb}) \]
\[ x_y, x_u, x_v \leftarrow \text{SPLIT-CHANNELS}(x_{yuv}) \]
\[ p_y, p_u, p_v \leftarrow \text{SPLIT-CHANNELS}(a_{yuv}) \]
\[ x_{yuv}' \leftarrow \text{MERGE-CHANNELS}(x_y, p_u, p_v) \]
\[ x_{rgb}' \leftarrow \text{YUV-TO-RGB}(x_{yuv}') \]
\[ \text{return } x_{rgb}' \]

3.3.2 Multiple Content and Style Transfer

Since the style and content images are encoded in the network, multiple style or content images can be used to compute the gradient. We can interpolate between the image representations during the initial feed-forward passes; the interpolation between the images is not done as a post-processing step. We can combine multiple style images and content images and control the degree of blending between the different style images or content images. We can transfer the style from multiple style or content images using:

\[
L_{\text{total}}(P, A, x, L^C, L^S) = \alpha \sum_{p \in P} w_p L_{\text{content}}(p, x, L^C) + \beta \sum_{a \in A} w_a L_{\text{style}}(a, x, L^S)
\]  

(3.14)

where $\alpha$ is the weighting of the content reconstruction, $\beta$ is the weighting of the style reconstruction, $L^C$ is the set of layers used in the content representation, $L^S$ is the set of layers used in the style representation, $P$ is a set of content images, $A$ is a set of style images.
3.3.3 Weighted Style Transfer

A limitation of the original style transfer algorithm is that the style is uniformly applied throughout the entire image oblivious to the content image semantics. In portraiture, this can cause human subjects to blend into the background. Chan et al. presented an extension of the style transfer algorithm to perform weighted style transfer [5]. By manipulating the Gram matrices, the style transfer can be weighted according to the semantics of the content image. Since the style images are scaled to the dimensions of the content image before the feed-forward passes through the network, a semantic segmentation of the content image can be used as a mask to weight the transfer of the style (Figure 3.4).

![Figure 3.4: Weighted Style Transfer](image)

A layer $l$ with $N_l$ feature maps of size $H_l \times W_l$ has an activation volume $V^l \in \mathcal{R}^{H_l \times W_l \times N_l}$. We define a function $mask$ to perform an element-wise multiplication of the scaled and normalized content mask $m \in \mathcal{R}^{H_l \times W_l}$ and each feature map in a layer $l$. That is, we expand the mask along the depth to the dimensions of $V^l$ and then perform an element-wise multiplication. We define the masking func-

23
where \( \otimes \) denotes an element-wise multiplication of tensors.

When including content masks to weight the style transfer, we define the style loss function as:

\[
L_{\text{style}}(a, x, m, L^S) = \sum_{l \in L} \frac{1}{4N_l^2M_l^2} \sum_{i,j} (\text{mask}(G^l, m)_{i,j} - \text{mask}(A^l, m)_{i,j})^2
\]

(3.16)

where \( a \) is the style image, \( x \) is the stylized image, \( m \) is the content mask, \( L^S \) is the set of layers used in the masked style representation.
A display connected to a digital computer gives us a chance to gain familiarity with concepts not realizable in the physical world. It is a looking glass into a mathematical wonderland.

- Ivan Sutherland, 1965

CHAPTER 4 describes the implementation details, experiments, and results of the techniques described in Chapter 3. Artistic style transfer is not a well defined problem but it can be viewed as a visual Turing test. This test requires that a human observer should be unable to distinguish a synthesized artwork from a real
artwork regardless of the content present. Therefore, our experiments focused on obtaining qualitative results where an experiment is considered successful if the synthesized artwork “looks like” the real artwork. We acknowledge this criterion of success is subjective. Qualitative results are presented for all of the previously described techniques. Quantitative results are presented for a subset of the described techniques.

4.1 IMPLEMENTATION DETAILS

The techniques described in Chapter 3 were implemented in the Python programming language using the TensorFlow and OpenCV libraries. TensorFlow is an open-source machine learning library currently maintained by Google [11]. OpenCV is an open-source computer vision library currently maintained by Itseez. Specifically, the TensorFlow C++ library was used through TensorFlow’s Python API and OpenCV’s C and C++ libraries were used through OpenCV’s Python bindings. Weights and biases from the pre-trained VGG-19 Convolutional Neural Network were used [8]. The pre-trained VGG-19 model can be downloaded from:

http://www.vlfeat.org/matconvnet/pretrained/

The Python source code was constrained to a single file to allow the project to be easily used and distributed by users of all skill levels. Usage scripts were created to help users unfamiliar with Python or the specific project details quickly install the required dependencies and execute the code. The program accepts 47 command-line arguments allowing advanced users to customize the behavior of
the program. The source code, a detailed README, and a collection example images are publicly available and free for use under the open-source GNU General Public License at:

http://www.github.com/cysmith/neural-style-tf

4.2 SINGLE-IMAGE STYLE TRANSFER

The main contribution of the style transfer algorithm is that content and style can be separated, manipulated, and combined to produce new synthesized images. We demonstrate this result by separating the content and style of two RGB images and then combining the content from one and the style from the other to synthesize a new RGB image (Fig. 3.1). For many of our experiments we used a collection of images that have become the de facto standard test images for the style transfer algorithm. The content image is a photograph of the Neckarfront in Tübingen, Germany. The style images are photographs of several iconic paintings in art history: The Shipwreck of the Minotaur by J.M.W. Turner, The Starry Night by Vincent van Gogh, Der Schrei by Edvard Munch, Femme nue assise by Pablo Picasso, Composition VII by Wassily Kandinsky (Appendix A.2). These images were used to compare our results to the results obtained by Gatys et al.’s Caffe implementation [1] and Johnson’s Torch implementation [12] (Fig. 4.1). Our results are qualitatively similar to the results obtained by Gatys et al. and Johnson. The subtle perceptual differences between the three implementations are likely due to using different parameters or images of the artworks.
Figure 4.1: Qualitative Comparisons of Implementations.

(left) Gatys et al. [1], (middle) Johnson [12], (right) Our Implementation
4.2.1 Optimization

The optimization algorithms L-BFGS and Adam were used to minimize the total loss. Gatys et al. used the L-BFGS optimizer [1] and Siegel et al. used the Adam optimizer [10]. Since the L-BFGS optimizer is not currently supported in Tensorflow an interface to the SciPy L-BFGS optimizer was used. We compared the convergence of the L-BFGS optimizer and the Adam optimizer using 3 different learning rates for the Adam optimizer. The total losses over 1000 optimization iterations are shown in Fig. 4.2. The quantitative results show the L-BFGS optimizer converges faster than the Adam optimizer. The Adam optimizer produces inferior qualitative results because of numerical instability. As seen in Fig. 4.2 when the learning rate $r = 10$ there is a sharp spike in the loss after 500 iterations. These spikes corrupt the appearance of the stylized image and unfortunately lead to poor qualitative results even after the loss has stabilized. The learning rate can be tuned to reduce numerical instabilities but this often requires many trial executions to determine a sufficient learning rate value for a given pair of images. Another drawback of tuning the learning rate is that a small learning rate will require many more optimization iterations to produce a desired result. For these reasons, L-BFGS optimizer is recommended over the Adam optimizer.

To gain an understanding of how each loss function converges, the intermediate results of the loss functions were compared (Fig. 4.3). The L-BFGS optimizer did not allow intermediate results to be saved, so we used the Adam optimizer to save intermediate results. Fig. 4.3 shows the convergence for the total, content, style, and total variation loss functions over 1000 optimization iterations using an
image of the Neckarfront as the content image and *The Starry Night* as the style image.

\[ L_{\text{total}} \] Convergence

\[ L_{\text{total}} \] convergence

\[ L_{\text{content}} \]
\[ L_{\text{style}} \]
\[ L_{\text{tv}} \]

**Figure 4.2:** L-BFGS and Adam Optimizer Convergence

**Figure 4.3:** Individual Loss Convergence
4.2.2 Relative Weighting

The relative weighting is the $a/\beta$ ratio where $a$ is the content weight and $\beta$ is the style weight in the total loss function (Eq. 2.8). Since the total loss function is a linear combination of the style loss function and the content loss function, the relative weighting determines the degree of the content or style in the synthesized image. That is, if the relative weighting is high (e.g. $1 \times 10^{-2}$) the content image will be clearly recognizable but the style may not closely resemble the artistic style. On the other hand, if the relative weighting is low (e.g. $1 \times 10^{-5}$ the content image may be obscured from recognition and the synthesized image will resemble the style image. Fig. 4.4 shows synthesized images for 4 different relative weightings $1 \times 10^{-5}$, $1 \times 10^{-4}$, $1 \times 10^{-3}$, and $1 \times 10^{-2}$ using an image of the Neckarfront in Tübingen, Germany as the content image and an image of the artwork Composition VII by Wassily Kandinsky as the style image. The third, fourth, and fifth rows most clearly show the content is more apparent with a high relative weight (e.g. $1 \times 10^{-2}$) while the style is more apparent and the content is unrecognizable with a low relative weight (e.g. $1 \times 10^{-5}$). Like Gatys et al. we also found the relative weight $1 \times 10^{-3}$ generally produces a desired result and can be used as a sufficient default value. Unfortunately, the degree of style existing in art varies between artists and artworks, so choosing a relative weight to match the style of the artwork requires some tuning. There is a trade-off between the content and the style in the reconstruction but this trade-off can be carefully controlled to produce a desired result.
Figure 4.4: Relative Weighting and Layer Representations

4.2.3 Layer Representations

The layers used to construct the style and content representation have a significant impact on the qualitative results of the synthesis. Features from separate
subsets of the convolutional layers are used in the style representation and content representation.

Gatys et al. recommend using features in the convolutional layers ‘conv1_1’, ‘conv2_1’, ‘conv3_1’, ‘conv4_1’, and ‘conv5_1’ for the style representation. The intuition behind these choices is the receptive field sizes and feature complexities of a Convolutional Neural Network increase down the depth of the network’s layered hierarchy. Therefore, Gatys et al. found that matching style representations at high layers in the network created the most visually appealing results. This increasing size and complexity of local image structures can be seen in Fig. 4.4. Gatys et al. also recommend using features in the convolutional layer ‘conv4_2’ to match the content representation. The intuition behind this choice is the image structures in higher layers of the network are less rigid or more malleable to alterations from the style representation; i.e. the style does not appear to be an overlay over the content.

4.2.4 GRADIENT DESCENT INITIALIZATION

The gradient descent can converge to different local minima. To achieve a desired result it is possible to bias the convergence by initializing the gradient descent with either Gaussian noise, the content image, or the style image. Initializing the gradient descent with the content image and style image seems to bias the result toward the spatial structure of the initialization image (Fig. 4.6a 4.6b). Initializing the gradient descent with Gaussian noise, the content image, or the style image does not seem to have a recognizable effect on the overall transfer of style

33
because the methods converge similarly. There are some advantages and disadvantages to each type of initialization. Initializing the gradient descent with the content image seemed to produce the best results by preserving the spatial structures of the content image. Initializing the gradient descent with the style image requires more optimization iterations to synthesize the content of the image. Initializing the gradient descent with an image of Gaussian noise generally takes longer (i.e. requires more optimization iterations) to converge to a desired result, but allows an arbitrary number of distinct images to be synthesized (Fig. 4.6c 4.6d). Whereas initializing with a fixed image deterministically converges to the same result (Fig. 4.6a 4.6b).

**Figure 4.5:** Convergence for Gradient Descent Initialization
MULTIPLE STYLE TRANSFER

To demonstrate the transfer of multiply artistic styles, we synthesize a new image using two style images and one content image. The degree of blending between the two style images is controlled by weighting each style image and normalizing the weights such that the sum of the weights is 1.0. This technique blends the two styles more convincingly than simply interpolating between two stylized images as a post-processing step. Fig. 4.7 shows several interpolations be-
tween the style of *The Starry Night* by Vincent Van Gogh and *The Scream* by Edward Munch. However, a limitation of this technique is the styles are blended non-deterministically and it is not possible to control the orientation or location of the blending.

![Content image](image1) ![7:3 style ratio](image2) ![5:5 style ratio](image3) ![3:7 style ratio](image4)

**Figure 4.7**: Multiple Style Transfer

### 4.2.6 Preservation of Color in Style Transfer

We transferred the color from the content image using color spaces allowing the luminance channel to be separated from the color channels; specifically, we used the YUV, YCrCb, CIE L*a*b*, or CIE L*u*v* color spaces. We compared the re-
results for each color space conversion. The results show there are advantages and disadvantages to preserving the color scheme of the original image using the luminance-only color transfer technique. An advantage of the luminance-only method is the colors of the content image are preserved exactly [3]. A disadvantage of the luminance-only transfer method is the dependencies between the luminance and the color channels are lost. The loss of these dependencies can cause the stylized image to have perceptually unrealistic characteristics such as brush strokes composed of many different colors [3].

Figure 4.8: Preservation of Original Colors
Figure 4.9: Color Space Conversions
The conversion to the content image colors can be performed before or after the synthesis. When performing luminance-only style transfer before the style transfer, we converted the colors of the style image to the colors of the content image before the synthesis. Likewise, when performing luminance-only style transfer after the synthesis, we converted the colors of the stylized image to the colors of the content image. Fig. 4.9 shows the results of transferring the colors of the content image before and after the synthesis using the color spaces YUV, YCrCb, and CIE L*a*b*. Overall, each color space produced similar results and successfully preserved the colors of the content image. The CIE L*u*v* color space resulted in a poor color conversion and should not be used for this technique.

4.2.7 **Texture and Photorealistic Transfer**

The style transfer algorithm developed from previous research using Convolutional Neural Networks to synthesize arbitrary textures. Since the style transfer algorithm is essentially a texture transfer technique, the algorithm is not constrained to only transferring artistic styles. The algorithm can also be used to transfer the style of textures. The synthesis of a photorealistic texture and a content image can produce pareidolic imagery.

To some degree, the algorithm can even be used for photorealistic style transfer where a high relative weight (e.g. \(1 \times 10^{-2}\)) is used in hopes of synthesizing a photorealistic image (Fig. 3.11). However, the noise due to the convolutional layer filters limits the effectiveness of photorealistic style transfer. Photorealistic style transfer also only seems effective when both images share similarities in
semantic content (Fig. 3.11).

Figure 4.10: Style Transfer Using Textures

Figure 4.11: Photorealistic Style Transfer
4.2.8 **Weighted Style Transfer**

A limitation of the original style transfer algorithm is the style is applied uniformly throughout the entire synthesized image without an awareness of the semantics of the content image. Deconvolutional Neural Networks have recently been shown to perform state-of-the-art results in semantic segmentation [13]. Semantic segmentation masks allow the style to be transferred to particular objects within the image. To demonstrate the use of semantic segmentation masks, we obtained portrait images and corresponding segmentation masks from [14].

Our results show qualitative improvements over the results obtained by Chan et al. [5]. The reasons for the perceptual improvements are likely due to a few differences between the implementations. We used the L-BFGS optimizer instead of the Adam optimizer. We used five layers 'relu1_1', 'relu2_1', 'relu3_1', 'relu4_1', 'relu5_1' in the style representation instead of the single layer 'conv1_1'. We applied the masks to the Gram matrices in multiple convolutional layers instead of a single convolutional layer. The mask images were scaled and down-sampled using bicubic interpolation rather than using a pooling layer. We believe scaling the mask image to the dimensions of the input image using multiple convolutional and pooling layers may yield better results.
Figure 4.12: Weighted Style Transfer

By using multiple mask images, different styles can be transferred to specific regions in the image. The masking is not done as a post-processing step; we are not simply masking two stylized images with a threshold. Since the mask is inserted into each style layer of the Convolutional Neural Network, the masking does not result in hard edges around the boundaries. Instead, the boundaries between the background and foreground blend together because the masking is performed during the synthesis of the image. Fig. 4.13 shows a weighted style...
transfer using two style images and two content masks. The soft blending between the foreground and background can be seen at the top of the image where the woman’s hair flows into the background. Since the blending is performed during synthesis, the edges around the boundary are smooth and more natural than simply masking two images after synthesis.

![Foreground Mask](image1.png) ![Stylized Image](image2.png)

**Figure 4.13:** Masking the Gram Matrix

### 4.3 Video Style Transfer

Transferring an artistic style to a sequence of video frames is a challenging problem. If the each frame of the sequence is processed independently the resulting video will have a noticeable amount of flickering and false discontinuities. If the gradient descent of each frame is initialized with Gaussian noise, the frames may
converge to different local minima causing flickering. If the gradient descent of each frame is initialized with the previously stylized frame, the transitions between video frames will be smoother but temporal discontinuities will cause flickering. To reduce the amount of flickering in the video, Ruder et al. [4] presented a technique of warping the previously stylized frame to the current frame using the estimated optical flow between the pair of frames. Additionally, a loss function for short-term temporal consistency was used.

For the experiments, we used video frames from the MPI Sintel Dataset [15]. The MPI Sintel Dataset provides video sequences from the film Sintel for the purposes of challenging researchers to improve existing optical flow benchmarks. Sintel is an open source animated short film produced by the Blender Foundation.

The parameters used for transferring artistic style to video were similar to the parameters used for single-image style transfer. We used the VGG-19 network for computing the losses. The layer ‘relu4_2’ was used to represent the content and the layers ‘relu1_1’, ‘relu2_1’, ‘relu3_1’, ‘relu4_1’, and ‘relu5_1’ were used to represent the style. Ruder et al. [4] augmented the Torch L-BFGS optimizer to consider the loss converged if it did not change by more than 0.01% or 0.1% over the previous 50 optimization iterations. We tried this approach but there was a wide variance in the number of optimization iterations for arbitrary frames, resulting in inconsistencies between the style transfer in each frame. Instead we specified the number of optimization iterations for the first frame and for each frame after the first frame. For the first frame, we used a default value of 3000. For each frame after the first frame, we used a default value of 1000. While this approach probably
takes longer, it was generally more stable than other criteria for convergence.

We performed style transfer on frames 1 to 50 from the MPI Sintel Dataset frame sequence 'bandage_2'. The resolution of the frames was $1024 \times 436$ pixels. We used DeepFlow [16] to compute the estimated forward and backward optical flow between frames. We to minimize the amount of flickering and false discontinuities in the stylized video. We compared the qualitative results when using the short-term temporal loss function and when using the different initialization techniques: Gaussian noise, the previously stylized frame, and the previously stylized frame warped to the current frame.

The qualitative results of the naive approach and the approach of incorporating a temporal constraint and initializing each frame from previously stylized frame warped to the current frame are compared in Fig. 4.14. Although it is difficult to present a comparison of flickering in a video sequence using static images, the use of the temporal constraint and warping according to optical flow shows a visible improvement over the naive approach. This result is obvious when the frame sequences are animated. The naive approach has a significant flickering while the other approach results in smoother transitions between frames.
Figure 4.14: Video Style Transfer
If what AARON is making is not art, what is it exactly, and in what ways, other than its origin, does it differ from the “real thing?” If it is not thinking, what exactly is it doing?

- Harold Cohen (on his AI painter AARON), 1995

5

Conclusion

5.1 Conclusion

We presented an analysis and implementation of several techniques for artistic style transfer: Single Style Transfer, Multiple Style Transfer, Video Style Transfer, Preservation of the Original Colors, and Weighted Style Transfer. Additionally, we presented two extensions of previous techniques. Both of these extensions
produced similar results to the previous approaches or improved upon the previous results.

During the course of this research, several significant extensions of Gatys et al.’s *Neural Algorithm for Artistic Style* were developed by other researchers. Although the *Neural Algorithm for Artistic Style* produces great results and high-quality images, the algorithm is computationally expensive. Feed-forward generator networks were developed by Ulyanov et al. [17] and Johnson et al. [18]. Ulyanov et al. showed that replacing batch normalization with instance normalization dramatically improves the results of feed-forward style transfer networks [19]. The feed-forward networks are orders of magnitude faster than the *Neural Algorithm for Artistic Style* but the perceptual quality of the style transfer is often noticeably reduced. Currently, there exists a trade-off between the speed and quality of the style transfer. Nonetheless, feed-forward networks were quickly incorporated into the viral mobile application Prisma. More recently, Dumoulin et al. [20] created an extension allowing feed-forward multiple-style transfer. Facebook demonstrated a real-time feed-forward network for mobile apps using Caffe2Go, a deep learning framework developed specifically for mobile phones. The popular appeal of artistic style transfer algorithms suggests there is a bright future ahead for the application of machine learning techniques to generate artistic imagery.
A.1 HARDWARE AND SOFTWARE SPECIFICATIONS

The exact hardware and software components used were:
<table>
<thead>
<tr>
<th>Hardware</th>
<th>Make &amp; Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Core i7-6800K @ 3.40GHz × 12</td>
</tr>
<tr>
<td>GPU</td>
<td>NVIDIA GeForce GTX 1080/PCIe/SSE2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>Ubuntu 16.04.1 LTS 64-bit</td>
</tr>
<tr>
<td>Python</td>
<td>2.7.12</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>0.10.0rc</td>
</tr>
<tr>
<td>OpenCV</td>
<td>2.4.9.1</td>
</tr>
<tr>
<td>CUDA</td>
<td>8.0</td>
</tr>
</tbody>
</table>
A.2 **Style Images**

(a) “The Starry Night” by Vincent Van Gogh (1889)  
(b) “Shipwreck” by William Turner (1805)  
(c) “Red Canna” by Georgia O’Keeffe (1923)  
(d) “Composition VII” by Wassily Kandinsky (1913)  
(e) “Seated female nude” by Pablo Picasso (1910)  
(f) “Light Iris” by Georgia O’Keeffe (1924)  
(g) The Scream by Edvard Munch (1893)  
(h) “Woman with a Hat” by Henri Matisse (1905)  
(i) “Self-Portrait” by Pablo Picasso (1907)

*Figure A.1: Style Images*
References


