7-2016

A Temporally Coherent Neural Algorithm for Artistic Style Transfer

Michael Dushkoff

A Temporally Coherent Neural Algorithm for Artistic Style Transfer

by

Michael Dushkoff

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science
in Computer Engineering

Supervised by

Dr. Raymond Ptucha
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
July 2016

Approved by:

Dr. Raymond Ptucha, Professor
Thesis Advisor, Department of Computer Engineering

Dr. Andreas Savakis, Professor
Committee Member, Department of Computer Engineering

Dr. Nathan Cahill, Professor
Committee Member, Department of Mathematical Sciences

Dedication

I would like to dedicate this thesis to my loving parents, Cathy and Vlado Dushkoff, whose unabating support in everything that I strive for has fostered my own indefatigable desire to discover and consistently push the boundaries of creativity.

I would also like to dedicate this research to my advisor, Dr. Raymond Ptucha, whose contagious energetic enthusiasm and passion for his area of research is what had originally inspired me to pursue the rapidly expanding field of Machine Intelligence.

Acknowledgments

I would like to thank my committee members, Dr. Nathan Cahill and Dr. Andreas Savakis, for their thought-provoking insights and suggestions that have invariably improved this work. I would also like to acknowledge the large number of contributions to this work by Ryan McLaughlin who had originally worked with me on publishing this research. Without his devoted, hardworking attitude and endless creativity, this work would certainly not have been possible.

Finally, I would like to acknowledge NVIDIA for donating the GPUs utilized in making this research possible.

Abstract

A Temporally Coherent Neural Algorithm for Artistic Style Transfer

Michael Dushkoff

Within the fields of visual effects and animation, humans have historically spent countless painstaking hours mastering the skill of drawing frame-by-frame animations. One such animation technique that has been widely used in the animation and visual effects industry is called "rotoscoping" and has allowed uniquely stylized animations to capture the motion of real life action sequences, however it is a very complex and time consuming process. Automating this arduous technique would free animators from performing frame by frame stylization and allow them to concentrate on their own artistic contributions.

This thesis introduces a new artificial system based on an existing neural style transfer method which creates artistically stylized animations that simultaneously reproduce both the motion of the original videos that they are derived from and the unique style of a given artistic work. This system utilizes a convolutional neural network framework to extract a hierarchy of image features used for generating images that appear visually similar to a given artistic style while at the same time faithfully preserving temporal content. The use of optical flow allows the combination of style and content

to be integrated directly with the apparent motion over frames of a video to produce smooth and visually appealing transitions.

The implementation described in this thesis demonstrates how biologically-inspired systems such as convolutional neural networks are rapidly approaching human-level behavior in tasks that were once thought impossible for computers. Such a complex task elucidates the current and future technical and artistic capabilities of such biologically-inspired neural systems as their horizons expand exponentially. Further, this research provides unique insights into the way that humans perceive and utilize temporal information in everyday tasks.

A secondary implementation that is explored in this thesis seeks to improve existing convolutional neural networks using a biological approach to the way these models adapt to their inputs. This implementation shows how these pattern recognition systems can be greatly improved by integrating recent neuroscience research into already biologically inspired systems. Such a novel hybrid activation function model replicates recent findings in the field of neuroscience and shows significant advantages over existing static activation functions.

Contents

List of Figures

Glossary

A

Artificial Neural Network A complex network of computational units called ”neurons” which are a mathematical abstraction of biological neurons, p. 15.

C

Convolutional Neural Network A breakthrough technique in the field of machine intelligence wherein network of artificial neurons are arranged such that individual layers perform a convolutional operation with other local neurons, p. 11.

E

Example-Based Rendering A non-photorealistic rendering technique that can produce outputs that look visually similar to a given artistic style, p. 7.

M

Markov Random Field A set of random variables described by an undirected graph wherein movement along the graph nodes is random,

p. 13.

Multilayer Perceptron A multilayer approach to an artifical neural network, p. 16.

N

Non-Photorealistic Rendering A technique of replicating specific elements of artistic styles such as brush strokes in order to replicate that style with respect to a new content image. These techniques usually focus on creating outputs that are not true to real life, p. 6.

O

Optical Flow The apparent motion produced by relative movement of objects in a scene to an observer, p. 51.

P

Principle Component Analysis An algorithm used to reduce the dimensions of a problem into its most important components, p. 57.

R

Rectified Linear Unit A non-linear activation function used in artifical neural networks, p. 16.

S

Stroke-Based Rendering A non-photorealistic rendering technique which focuses mainly on brush stroke reconstruction, p. 7.

Structured Edge Detector A new fast edge detection algorithm, p. 56.

Chapter 1

Introduction

1.1 Problem Definition

A typical animator using modern computerized drawing tools can produce several seconds of animation per week, the exact amount dependent on the scene complexity, motion, dialogue, and fidelity of the required animation. Dozens of animators may even work for multiple years on a motion picture, spending much of their time on detailed animation work that focuses on frame-by-frame stylization. As with working on a project with many people, the individual styles and artistic capabilities will differ between these people, which necessitates others to inspect the overall continuity of the work to meticulously create a seamless production. Tools which can either speed up this animation process, normalize drawings across animators, or apply similar styles to all drawings are areas of active research and are quite difficult tasks for a computer to tackle which is why currently most animations are still done in the laborious fashion.

The complexity of this problem is quite evident as it is even a non-trivial task to humans, which necessitates its punctilious decomposition in order to

allow the problem to be properly addressed. As such, the problem can be broken down into two main components. The first component addresses the artistic stylization of a piece of work to resemble a chosen aesthetic, while the second component deals with allowing this stylization to be coherent over time to produce natural and visually appealing animations. The assessment of these traits is often subjective and requires human input in order to determine the overall quality of an algorithm that is used to automate this process. The separation of these two tasks can allow the problem to be more easily addressed in smaller more manageable pieces.

1.2 Contributions

The main contribution of this thesis includes a relatively new method to create artistically stylized videos using any input artistic style imaginable and whose output frames are temporally similar to the input motion [21]. These outputs are free from distracting jitter artifacts and shower door effects which are inherent in older techniques that do not take into account accurate motion tracking along with stylization. This new technique combines recent dense optical flow techniques to a neural network-based approach of artistic stylization in order to produce compelling and interesting artistic videos that faithfully preserve the motion of textures in objects in a scene over a period of time. The technique itself presents a novel way of decomposing temporal information in a scene that integrates the information inherent in the hierarchical structure of a convolutional neural network

into the time domain. Insights that can be gained from this technique are emergent in the fields of computer vision and machine intelligence and are compared to recent neuroscience research findings.

The intermediate contributions of this paper include a novel methodolgy for training neural networks using multiple adaptive activation functions per neuron in order to increase classification accuracies on complex standard datasets [22]. It is with this improvement that an overall more biologically feasible model for neural computation was produced that improves the robustness, accuracy, and computational resources of modern-day artificial neural networks. Various techniques were explored including a novel hybrid activation function model that seeks to replicate recent findings in the field of neuroscience, as well as the introduction of a new algorithm for optimizing and training non-monotonic activation functions. The significant improvements that stand out from these model-level adaptations are faster training times when compared to traditional static convolutional neural network models and the possibility of reducing the memory required to allow these models to train to the same accuracies.

1.3 Organization

The organization of this thesis will be as follows: in order to address the first problem of artistic stylization, a history of previously utilized rendering techniques will be explained thoroughly in order to build a substantial

background case for the newer methods used. Among these sylization techniques is a new technique based in a biologically inspired technology which has been recently revived in the last decade within the pattern recognition community referred to as neural networks.

This concept of computerized artificial neural networks will be introduced and thoroughly explained in order to more properly allow the method of artistic stylization to be increasingly understandable. Within this section, a subset of experiments will be described about which these mathematical abstractions of the human brain can be improved upon. Among these abstractions, a newly developed hybrid activation function method will be analyzed thoroughly to show the improvements that the current generation of neural networks stand to benefit from. The results of such experiments will elucidate the important improvements on current generation artificial neural systems that can be gleaned from the recent discoveries in neuroscience research.

Next the concept of temporal coherence will be introduced. This will include a discussion of various optical flow techniques and how they have improved over time. A few different examples of these algorithms will also be presented for the reader to understand their differences. Among these methods, a newer dense optical flow method will be more thoroughly explained with relation to temporal coherence in video frames. The methods will all be thoroughly analyzed for their strengths and weaknesses in the area of providing this temporal coherence including the average estimated

flow error and the computational complexity of the overall algorithm.

After the previous sections have been covered, the reader should have a more complete understanding of the concepts involved in the underlying methodology. The full methodology will be broken into two sections: stylization and temporal coherence just as in the previous sections. These sections will go into complete detail on how the video stylization algorithm was conceptualized, describing the specific optical flow method utilized based on the comparisons of the various methods.

Finally, the results will be shown of the produced algorithm and compared extensively to various different improvements. Due to the nature of the algorithm, links to the full video results will be provided for the reader to accurately compare and judge their quality. A few conclusions will be drawn about these results in relation to recent advances in the field of machine intelligence and neuroscience which will complete the overall understanding of the impact that this research will leave on these fields.

Chapter 2

Artistic Stylization

2.1 Stylization Techniques

Over the past few decades, the advent of computational systems has revolutionized innumerable fields. Specifically the area of Artistic Rendering has expanded in the wake of advanced computerized graphics techniques in the early nineties. A new field of artistic expression referred to as Non-Photorealistic Rendering (NPR) quickly exploded into an active research area which spanned a plethora of different styles and forms of visual communication [55]. Much of this research was dedicated to producing a creative or artistic tool that can significantly enhance the way that artists produce their works in the way of aesthetic quality and diversity. Primarily, many of these NPR techniques focus on specific aspects of artistic rendering such as reproducing physical brush strokes, pen and ink strokes, regional style characteristics, textures of objects, and other complex patterns. Naturally the taxonomy of these techniques is broken into several distinct categories that encompass different methodologies of producing artistic stylization. The relative research interest in each category of NPR is shown below

in Fig. 2.1.

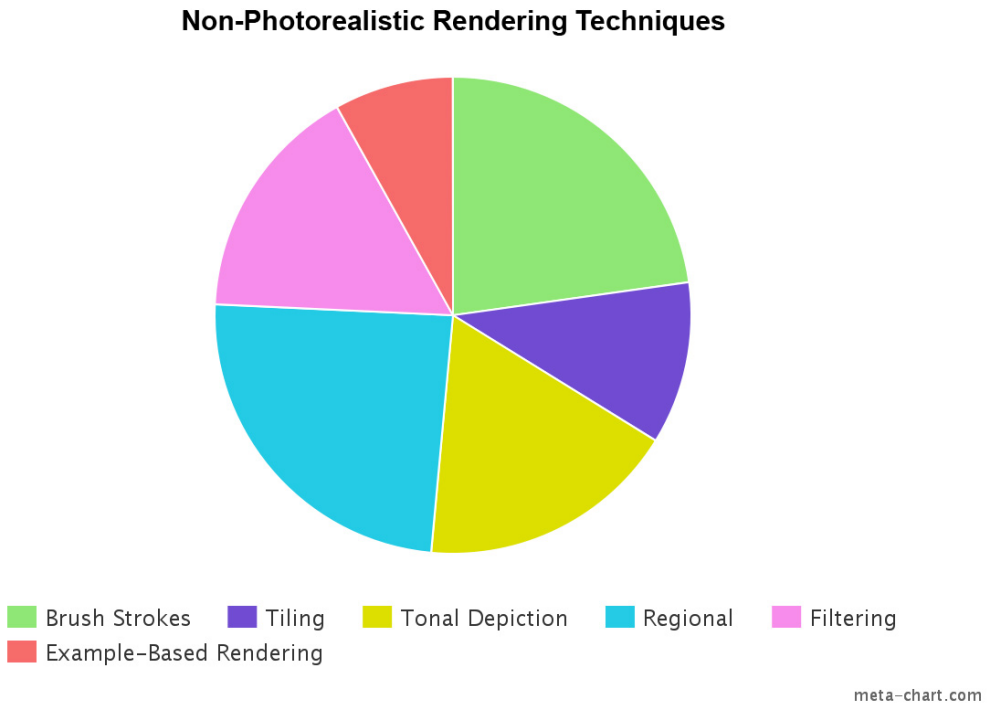


Figure 2.1: Artistic Rendering Techniques (Relative percentages based on [55] from data collected up until 2013).

Most of the research has been dedicated to stroke-based and regional techniques, however a relatively new and emerging field of artistic rendering is that of Example-Based Rendering (EBR). While other techniques only seek to reproduce specific facets of artistic works, EBR can replicate the specific stylization of any type of art. Such a complex problem has only recently been fueled by advances in research and technology.

2.1.1 Stroke Based Rendering

Stroke-Based Rendering (SBR) was an early form of image stylization that attempted to cover a 2D canvas with atomic rendering primitives according

to some process designed to emulate a very specific style [55]. Such early algorithms sought to virtualize the various diverse local artistic media primitives including brush strokes, tiles, stipples, and hatch marks. The process of covering a canvas with these primitives in order to render a given image in such styles was divided into two sub-categories referred to as local stroke algorithms or global stroke algorithms.

Local methods would perform decisions of stroke placement and angles based on the pixels within a specific spatial neighborhood, often guided by a user [36][37]. The utilization of straight rectangular brush strokes of varying sizes, oriented via Sobel gradients, were sufficient to generate aesthetically pleasing images which were directed by a user's movement of a virtual brush [36]. Litwinowicz et al. in 1997 detailed a fully automated version of this process using the same principles which clipped strokes based on Sobel edges [62]. The incorporation of a hierarchy of curved brush strokes and layered strokes according to size further improved the clarity and aesthetic quality of these techniques [42]. Layering strokes according to their size and relevance to the original image helped to mitigate the addition unnecessary overlapping of strokes that require no further detail [42].

Global methods on the other hand would optimize the placement of all strokes in an image such that they match realistic stroke placements. These algorithms brought a plethora of advancements in optimizing the retention of visual detail using Sobel gradients with regard to brush strokes [44][43], evolutionary algorithms [15], and snake relaxation [1]. Higher levels of

sophistication and finesse could be produced using global optimization As with local algorithms these came in the flavor of both semi-assisted and fully automated. As time went on, there was an increasing emphasis on such fully automated systems as well as a desire to extend such methods to video media. Many such algorithms related to these early stylization algorithms were produced at nearly the same time in the late 2000's, however these algorithms will be further discussed in Chapter 5.

2.1.2 Example Based Rendering

In contrast to SBR which focuses on encoding a set of heuristics to emulate a single particular artistic practice, a more widely representing method of stylization that can be more general in its application was also sought out. From this desire to encompass many different possible artistic stylization techniques, the notion of Example-Based Rendering (EBR) was born, largely pioneered by Hertzmann et al. who had produced many different AR techniques in the past [42][43]. This particular branch of AR compares a given image and an exemplar artistic style image in an attempt to transfer this style onto many different images. This would exponentially expand the artistic possibilities due to the fact that not only a single aspect of a piece of artwork could be emulated, but any aspect could be reproduced from a given artistic style in an amalgamation of that style and a new image.

Two separate branches of EBR exist which focus on particular aspects of

rendering that could be reproduced given an example image. Color transfer EBR and texture transfer EBR. The latter seeks to reproduce individual colors of an exemplar piece of artwork by matching approximate regional color histograms between the two images [68]. Similar methods have been implemented by replicating the mean and variance along the three channels in the CIELab color space as in [75], and by optimization of gradient and histogram matching which preserves perceptually vital local intensity differences, as in [92]. These outputs do not really capture local strokes and other subtleties of an artistic work and thus, the majority of early EBR techniques were focused on texture replication.

Among these early texture-replicating EBR techniques was a method referred to as "Image Analogy" [44]. This early technique combined machine learning and AR to reproduce artistic textures from examples by learning a mapping from a source image to its corresponding artistic depiction. This mapping can then be utilized to transfer texture information between images utilizing patches of textures by finding the closest matching pixels between the unaltered example image and the to-be styled test image using an approximate nearest neighbor algorithm. This mapping of the styled example image would maintain coherence with nearby pixels in the styled test image [44]. The caveat of this method is that two images must be utilized in the mapping process: one that is the unstylized example, and the other an artist's rendition of that original image. A significant limitation of many EBR methods is this utilization of the same image with and without

styling, which severely reduces the domain of artistic images that can be used as examples. Furthermore, many of these techniques rely on copying directly from the source artistic image patches of textures and then minimize global differences within the output image using some smoothness constraints. Nonetheless, EBR algorithms have attracted significant attention throughout the last decade and have even produced some photorealistic extensions. Such class of algorithms soon gave inspiration to researchers who wished to utilize the rapidly expanding field of machine learning to enrich the scope and transfer abilities of stylization algorithms. With the recent rise in interest in machine learning and in particular the revived field of neural networks, the artistic possibilities from such advances were fresh.

2.1.3 Neural Art

The most recent technique for stylized example-based rendering is classified as an EBR technique, however it can not be split merely into the traditional color or texture categories. This algorithm employs a more modern approach to the problem of automatically transferring artistic style to images which utilizes Convolutional Neural Networks (CNN)s or convolutional varieties of ANNs, which have currently surpassed human level performance in object recognition tasks, in order to generate artistic images that are stylized based on a reference image [40]. CNNs are a type of artificial neural network algorithm loosely based on the human visual system that utilizes convolutional operations to classify, segment, and even generate objects.

The novelty of this approach to transferring artistic style to content images comes from the application of the hierarchical layers of abstraction inherent in CNN architectures to separate and optimally combine content and style from two independent images [29]. The employment of CNNs in image recognition provides the ability to detect and differentiate massive varieties of features: edges, object classes, brush strokes, color, and shapes, among many others. CNNs abstract these features, or details about them, into hierarchical layers that build incrementally on one another wherein those pertaining to content and style can be separated [24][95][29].

Leon Gatys et al. [30] originally leveraged this hierarchical abstraction in order to produce generative textures from an example image without actually copying and stitching the textures as in traditional texture-centric EBR algorithms. The move away from this copy and stitch technique comes from the fact that styles are maximally represented not using the original image as a reference, but the Gram matrix of the original image's network activations. This places the possible solutions to the texturing problem on a plane of a multitude of minima which fundamentally abstract the possible texture minima that can be reached. This later became the basis of a new neural algorithm that utilizes the same abstraction capabilities of CNNs to produce new works of art given only an example artistic style [29]. This algorithm derives an optimal balance of the information stored in this neural-level hierarchy relating to the content and style of images in which they show that these aspects are indeed separable. As such, they were able to produce

images that optimally preserve the content of the original image, and the style of the artistic work all completely automatically through the naturally formed abstractions inherent in CNNs.

Recent work augments the neural-style algorithm by replacing the usage of Gram matrices with Markov Random Fields (MRF)s for style reconstruction [60]. Generation of the Gram matrix maintains the correlation of the filter channels, sacrificing local spatial information, however the use of MRFs can preserve much more of this local detail [60][13][29]. Conversely, MRFs have demonstrated an aptitude for preserving local spatial information while requiring significant effort to incorporate global spatial information [60][98][57][23]. This nature of MRFs make it an ideal combination with CNNs, providing both high level spatial abstractions while maintaining a higher fidelity of texture-level information.

The addition of semantic segmentation to these algorithms in [60] produced remarkable results, demonstrating that as with prior NPR methods like SBR, regional stylizing can be used with significant benefit to the aesthetic appeal [13]. A further interactive spin on this algorithm was produced wherein a user can segment an original artwork and draw new original works of art that reproduce stylizations within those specific boundaries and continuous in the overall artwork [13]. Such an interactive neural-based stylization technique can render very simple doodles into complex works of art that resemble famous artistic works [13].

Current research focused in improving this area of AR is ongoing as only

the surface of possibilities has been skimmed taking advantage of the rapid advancement of the machine intelligence and computer vision fields. As such, currently only a single video extension of this technique exists [78], however it should be noted that it was implemented after this work and does not fully address the points that we make. It is imperative to understand the underlying methodologies responsible for advancing this field in order to utilize such emergent technology to further improve and broaden the impact that it can have on problems that were once thought of as too complex or abstract for computerized systems to reach human-level proficiency in.

Chapter 3

Neural Networks

The human brain is made up of a network of billions of computational units, called neurons. The weighted sum of simultaneous inputs to a neuron determines its output spiking frequency which is subsequently passed on to other neurons. Each neuron is connected with up 10,000 other neurons, creating a network of around 100 trillion connections referred to as synapses. Artificial Neural Networks (ANN)s loosely mimic a simplified version of this biological network of connections digitally and have been typically implemented in software, but recently in hardware as well [85]. Breakthroughs in the arrangement and operations of these artificial neurons have made pattern recognition tasks even more viable for computers to solve within the past decade and have even allowed computers to surpass human recognition skills [40].

3.1 Background

The concept of ANNs began in the early 1940's as a mathematical abstraction of how neurons in the brain fired and the concept's evolution along the

next several decades brought incremental and rapid progress. An early pattern recognition algorithm based on ANN concepts was produced by Frank Rosenblatt in the early 1950's. This algorithm was referred to as the *perceptron* algorithm and was based on the perceptron model's fairly simple characteristic equation shown below in (3.1).

$$y = \sum_{i} (f(w_i x_i)) \tag{3.1}$$

This model mimics the way that biological neurons fire in response to a weighted sum of stimuli $w_i x_i$ based on a specific activation function f. Each perceptron applies its activation function to this weighted sum of inputs to allow the node to learn complex non-linear behaviors. This traditionally monotonic function restricts the output range of the neuron using a sigmoidal or tanh function[38]. Many other activation functions have been studied, but the Rectified Linear Units (ReLU)s [66], which clamp the negative outputs to zero and let the positive outputs go unchecked, have been the most successful as of late [33]. This artificial neural network model was originally limited in what patterns it could discriminate until the late 1970's when a breakthrough algorithm called *backpropagation* finally answered the question of how to train multiple layers of these perceptrons. The more complex models were referred to as Multilayer Perceptrons (MLP)s. These layered models consisted of an input layer, hidden layer, and output

layer wherein the input layer learns some higher order function of the linear combination of its inputs to assist classification as shown in Figure 3.1. The number of nodes in each layer, regularization strategies, addition of recurrent and skip forward connections, and network topologies have been studied in great detail in recent years [65]. These improved models created a resurgence in the machine intelligence community at the time, however they quickly dropped out of favor to simpler models such as support vector machines.

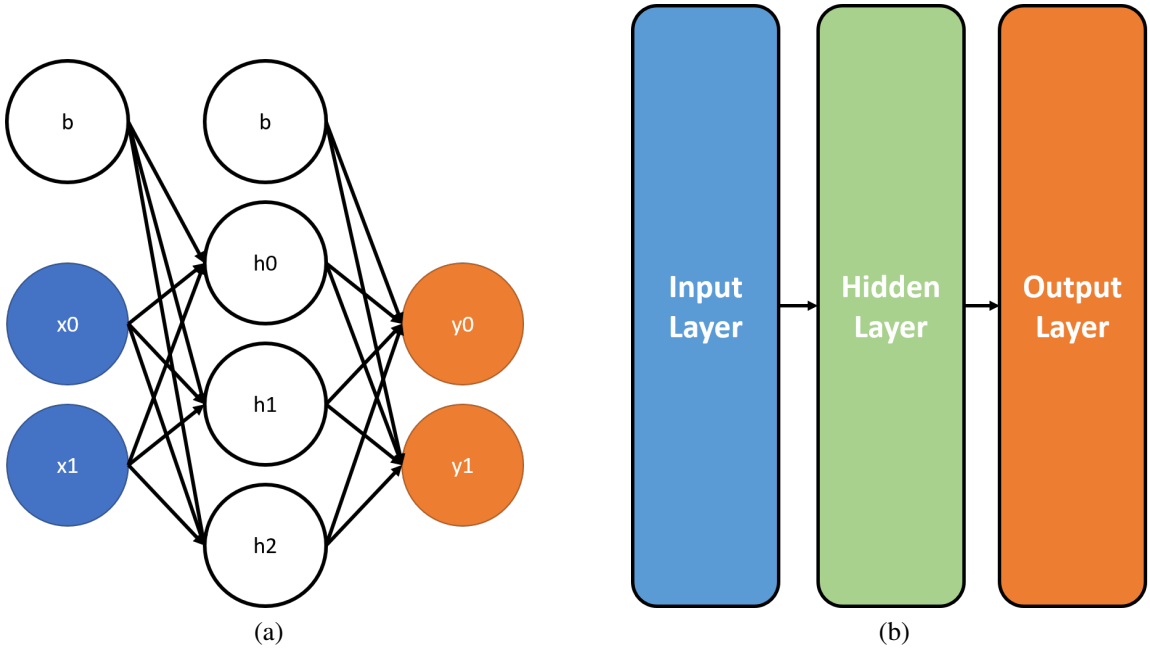


Figure 3.1: Basic ANN architectural organization. (a) Connectivity diagram for an MLP. (b) Corresponding layer-level descriptions.

In the early 2000's, ANNs had another key resurgence with the advent of deep learning and improvements to processing speeds. General purpose graphics processing units also spurred this resurgence and improved the

speed of computation of the massively parallel operations of these models. Deep learning was further fueled in 2011 by an early adaptation proposed by Yann LeCun et al. in 1995 [56] that applied a bank of learned convolutional filters at each layer instead of the traditional fully connected approach of MLPs. These networks were referred to as Convolutional Neural Networks, however at the time they were proposed, the computational complexity of these models prevented their favorable usage. With generations of improvements to computational systems, CNNs now prove to be much more robust models, reducing the overall connectivity while allowing deeper networks to be produced and trained. This breakthrough in the machine intelligence community is still ongoing as these models have continually shown cutting edge performance in a wide variety of pattern recognition tasks.

3.2 Activation Functions

In the late 1980s when neural networks were first gaining presence in the scientific community, Kurt Hornik described mathematically that any feed-forward neural networks with a single hidden layer connecting some finite number of neurons would be considered a universal approximator [47]. This property further showed that the multilayer perceptron type of artificial neural structure could theoretically approximate any continuous function given enough hidden neurons. Hornik also showed that it was not the actual activation function that gave these models this property, but rather the structure of the neural network itself. From this astounding conclusion, the ability

of these biologically-inspired models to be applied to a wide variety of applications was confirmed. Although Hornik had proved that the structure of the network caused this universal approximation capability, he did not, however address the practical implementation side-effects of ignoring the choice of activation functions. These concerns include the actual number of hidden nodes that are required for any specific task, the training time of the network, and even the feasibility of constructing a large enough network at the time. From these implications, it is certainly important to consider the impact that the specific choice in activation function has upon the network architecture in terms of training time and memory requirements.

Researchers realized these important implications of Horniks theorem and subsequent research investigated more complex activation functions to solve the problem of increasingly large network structures. Adaptive activation functions were explored as a means of achieving smaller network sizes. One of the first implementations that introduced the concept of adaptive activation functions that were applied to neural networks was developed in 1992. This first implementation was based on an adaptive polynomial function [71] which optimizes its own parameters to reduce the overall error of the network. This successfully reduced the model complexity of the network while simultaneously increasing the computational complexity of the activation function itself. One of the driving forces for this implementation was to create a hardware-feasible model that would perform comparably to sigmoidal function networks on various regression tasks.

The second adaptive implementation was based on a parameterized cubic spline function [12] which allowed the tradeoff of network complexity for activation function complexity. This implementation introduced the concept of learning parameters to optimize the activation function of a network as well as the network itself using the standard backpropagation algorithm. The spline itself was a Catmull-Rom based curve with a continuous first derivative which allowed backpropagation to work properly. For each of the control points of the spline curve, there are four other points that control the shape of the curve which are fixed in order to maintain sigmoidal properties of the overall function. This network was trained on a few small datasets using a variable number of neurons and outperformed the sigmoid function using less neurons. This clearly demonstrated the power of adaptive functions at an early stage, however the number of neurons within the network was a very meek 3 to 9 neurons [12].

A few follow-on papers were published regarding the adaptive cubic spline activation function including one that more concretely defined the theoretical capabilities of adaptive function network [87]. The spark that was ignited by the previous adaptive function research had necessitated the formulation of some type of formal analysis of the properties of such functions and their drawbacks. It was described that the large number of free parameters in the cubic spline method introduced increased difficulty in the learning process which subsequently removes the smoothness of each neurons output [87]. The large number of control points also inherently led

to overfitting of the training samples. In order to control this overfitting, the control points were spaced uniformly along the x-axis and only the y-dimension of these points were then allowed to vary. The smoothness of this function was then controlled by a penalty term which added additional error to adjacent points with very large distances in the y-dimension [87]. This ensured that the adaptive spline function that was learned had smooth increases and decreases, enabling easier optimization. Multiple networks were tested on various datasets and it was concluded that the restrictive properties added to the existing cubic spline adaptive function method had introduced a natural regularization that increased the generalization performance on many tasks [87]. Not only was the speed of convergence increased, but also the overall error was significantly lower with less free parameters with respect to the baseline sigmoidal networks.

Follow-on research [35] of the adaptive cubic spline function performed a critical analysis of these functions when applied directly to multilayer perceptron networks and for the first time tested their abilities on non-trivial problems. This research further concluded that the speed of convergence was significantly improved by these adaptive functions. Later implementations and architectures have mirrored these findings [96][51][93]. In 2003, research [28] analyzed the primary somatosensory cortex of a rats brain, revealing that biological neurons adapt their individual activation functions to deal with changing stimuli. Later on in 2004, a biological understanding of the desire for adaptive functions came to light. This paper [79] delved

into the underlying neurological processes that govern cell behavior that is crudely modeled in artificial neural networks. Specifically, the latent nature of memories was explained by the inherent modification of the activation function within single biological neurons which had been observed to have some sort of frequency dependent activations [79]. This difference in threshold of excitation causes neurons to react differently to stimuli with varying latencies, which allows the relation of time-dependent information. This finding spurred some further development of adaptive functions within artificial neural networks [93], however none of these implementations had decided to optimize their activation functions at a node-wise scale to mirror the adaptations that occur within individual biological neurons. This was most likely due to the fact that the technology available at the time would be prohibitively restrictive to such immense computation. Further developments in adaptive functions fell out of favor for a long duration of time afterwards.

Neural networks of the early 2000s were still not entirely sophisticated and were continually competing with simpler solutions such as linear regression and support vector machines, but with the advent of deep learning in the late 2000s to early 2010s, a fervent renewed interest in neural networks developed quite rapidly. This was a direct product of vastly improved computational capabilities as well as improvements in biologically inspired concepts which plunged the machine learning community into a research

tsunami that was created from this meteoric rise in the competency of artificial neural networks. Current generation research in the convolutional varieties of these neural networks have been producing state of the art results on datasets stemmed from the breakthrough performance demonstrated by Alex Krizhevsky and Geoffrey Hinton in 2012 in their paper that popularized such systems on the monumentally large ImageNet dataset [54]. From a biological standpoint, the next direction in applying the pre-existing early adaptive functions would be to allow individual neurons to differentiate their own adaptive functions. In concert with modern convolutional neural network architectures, this paradigm of adaptive activation functions hopes to have noticeable beneficial properties to the overall generalization capabilities of the network. This thesis seeks to take the next logical step in the research of adaptive functions when applied to modern-day computer vision problems, using current generation concepts in the field of deep learning.

A rather recent renewal of the interest in adaptive activation functions was spawned after the explosion of interest in deep learning and convolutional neural networks. This method introduced a small adaptive modification to the popularized ReLU activation functions that were previously producing record-breaking accuracies in convolutional neural network tasks [33]. This adaptation allowed the negative region of the ReLU function to decline at a learned slope [40]. This slope was learned on a layer-wise basis as part of backpropagation and the entire network architecture was

then tested on the ImageNet dataset. Along with some new initialization schemes and training implementations, the described architecture with these so-called Parametric Rectified Linear Units (PReLUs) surpassed human-level classification accuracy on the task of image classification [40]. This was an astounding accomplishment for such a simple tweak to the overall activation function of the network, which demonstrates the power that adaptability affords to already well performing convolutional neural network architectures.

Another modern paper that was recently released before this work has implemented a method for adapting activation functions on a node-wise basis for convolutional neural networks [2]. This method implements a piecewise-linear function that has variable parameters to tweak the position of the individual pieces and slopes of the linear segments that make up the overall activation function. This type of function is merely a sum of hinge-shaped functions that results in a fairly simple to compute output as well as derivative, however the function itself is not entirely differentiable. Each node within the network learns its own parameters to adjust this function to suit its needs independently in order to reduce the overall error of the network [2]. They compare their implementation to previous approaches with nonlinear activation functions such as maxout [34] and the rectified linear unit function [33] which have been performing well on classification problems in the past. This particular technique was used in conjunction with many training strategies on three separate modern datasets and obtained

state of the art improvements on all of them [2]. In all cases there was a significant improvement over the baseline case of non-adaptive functions for the same network architecture that was being tested on each dataset. One of the more limiting factors of these findings though is that there are only a fixed number of parameters that can be adjusted to tune the overall activation function for each node and the average spread of activation functions that were produced per layer were somewhat clustered with a fixed variance. This seems to suggest that the stringent conditions placed on these functions caused most of the functions to favor having a large number of very small varying activation functions in each node. A more interesting approach that will be explored is to utilize both non-linear adaptive monotonic functions as well as non-monotonic functions which have biological feasibility [16].

Non-monotonic functions have been explored extensively in early neural network implementations, mostly due to their increased discriminatory properties. In the classic XOR problem, a single node network utilizing a non-monotonic function would be sufficient in accurately discriminating all points, while in the classical monotonic implementation, two or more nodes are required. A multitude of papers from the late 1980s to early 1990s investigated non-monotonic functions for these desirable approximation capabilities. One such non-monotonic function of particular interest was the proposed Radial Basis Function which was found to have very powerful approximation capabilities that were later applied to neural network structures [74]. Research into radial basis function networks revealed that they

had universal approximation properties with single hidden layer multilayer perceptron networks [69]. This finding further showed the advantage of these networks for discriminating multidimensional inputs that are highly nonlinear in nature, thus showing their potential in applications of pattern recognition and classification.

A separate doctoral thesis on these non-monotonic functions as they are applied to multilayer perceptrons arose in 1993 [27]. This extensive study on non-monotonic activation functions revealed some very important insights about their training properties and also introduced a new family of functions called hyper-hill functions which were proven to be vastly superior to radial basis function networks [27]. This architecture required a novel training technique to surpass the learning speed and representation power of radial basis functions. One of the main limitations of this network architecture was the fact that hyper-hills are more prone to get stuck in local minima when compared to monotonic activation functions [27]. Further, these functions are very sensitive to the choice of learning rate and momentum applied during the learning algorithm which can very negatively affect the rate of convergence of the overall network [27]. Such problems can be addressed by exploring alternative training strategies for these non-monotonic functions which employ adaptive techniques for handling the nature of these functions [17].

In the late 1990s, a method for training non-monotonic function networks

materialized. This paper illustrates an approach that builds on the Barzilai-Borwein learning rate adaptation rule [73] which changes the learning rate to adapt to the gradient of the error surface. Furthermore, this method utilizes the non-monotonic Armijo line search acceptability criteria in order to scale the learning rate based on whether or not the next error is decreasing with respect to the maximum error encountered [73]. This ensures that the overall error can either decrease or increase within M iterations of the algorithm while escaping local minima. This training algorithm was successful in preliminary tasks and later was evaluated on pattern recognition tasks with impressive speed and accuracy when compared to normal backpropagation techniques [72]. This thesis will further explore these techniques as applied to modern-day problems in order to explore the possibilities of adaptive non-monotonic activation functions in convolutional neural networks.

The amalgamation of adaptive activation function techniques with non-monotonic functions hasnt been fully explored. Previous research shows that these two strategies separately have been proven to have promising capabilities in classification and regression tasks in terms of training times and resource utilization. These techniques are not, however without drawbacks and the method of training such networks that employ these approaches must be carefully considered. There is very little doubt that a more in-depth study of adaptive functions as applied on a node-wise basis to modern convolutional neural networks is integral to the understanding of how these networks can be further enhanced to handle modern-day machine learning

problems. This thesis seeks to bridge the gap in this current exciting field of machine learning research which has not fully explored these biologically inspired techniques.

3.3 Neural Network-Based Image Synthesis

Convolutional neural networks have arguably surpassed human level performance in object recognition tasks and have proven useful for various tasks in the pattern recognition community [39][83]. Traditionally these networks are trained in order to classify objects into categories, but they have also been used for object detection and segmentation [40]. An interesting property of these CNNs is that they systematically abstract images into a hierarchy of features of increasing complexity. This property has only recently been explored in more generative applications. Gatys et al. [29] utilize these CNNs in a generative manner to automatically transfer a reference image's artistic style to still images which was a method built from a CNN based texture synthesis technique [30]. The novelty of their method to transferring artistic style to content images comes from the use of the naturally occurring hierarchical layers of abstraction inherent to the way CNNs process images. This method leverages CNNs to extract and optimally combine content and style from two independent images. Convolutional layers of CNNs represent collections of filters activated by convolving these filters with their inputs. The objective of CNNs in image recognition is to detect and differentiate varieties of features including edges, textures, and parts of

objects among many others. CNNs abstract these features along a hierarchy of layers of increasing complexity and abstraction. Reconstructions from each layer's filter maps provide insight into the type of information they detail. At early layers, localized information becomes more important down to the level of individual pixels. In contrast, higher layers within a CNN increasingly represent objects and their global arrangement.

This unique hierarchical abstraction aspect of CNNs allows the optimization of style and content detail from pixel level information in early layers, to objects and global arrangements in later layers. The image to be generated can be optimized to maximally represent its original content with varying amounts of induced style. Further, the applied style constituent components of high, medium, to low frequency detail can be precisely controlled by changing the CNN optimization function at each hierarchical layer.

Conversely, at earlier layers, reconstruction shows a disregard for global information in favor of pixel-wise information retention. Hence the CNN hierarchy naturally separates content from style, where the global arrangement of objects is considered the content representation, and less abstracted regional information in lower layers is the style representation. Equipped with feature responses corresponding to style and content of separate images, the neural algorithm detailed in [29] produces an image optimally activating both, yielding an image with content of one image, and the style of the alternate image.

Chapter 4

Adaptive Methods

4.1 Activation Functions

In models of the human brain, the activation functions within each neuron transfer the sum of all incoming synapses to an expected firing rate [18]. These activation functions can be symmetric or antisymmetric as they exhibit excitatory or inhibitory functions in the brain. Whenever a neuron becomes active, the concentration of ions on the cell's surface will change as well as the concentration of ions within the cell [79].

When neural networks were first gaining presence in the scientific community, Kurt Hornik proved mathematically that any feed-forward network with a single hidden layer containing a finite number of neurons could approximate any continuous function [47]. This theorem showed that neural networks are inherently universal approximators and that the activation itself does not give it this property, but rather the inherent structures of a feedforward network architecture. This theorem, however does not take into account the number of hidden units that would be required to approximate any function, nor does it take into account the feasibility of training such

a network. From these implications, it is certainly important to consider the impact that the choice an activation function has on a specific network architecture's training time, and memory requirements on today's hardware.

Rectified linear units have been gaining popularity especially in deep convolutional neural network architectures. This non-saturating nonlinear function trains faster than saturating nonlinear functions [66][54]. By allowing the positive values of the output of a neuron to grow unbounded, the resulting output of each convolutional layer is allowed to exhibit intensity equivariance [66] such that scaling intensities within regions of the image will not change the overall decision capabilities of the network. These qualities are important in classification problems that depend on inputs being invariant.

This section of research analyses the benefits of complex activation functions on larger deep networks. Each neuron in the deep network is configured to allow any number of activation functions, whereby the turning on and off of each activation function is learned during the training process. A new deep learning library built in the Torch7 framework [14] incorporates the newly introduced activation functions and learns the weight parameters automatically during training. This new library allows nodes in a network to use a family of activation functions simultaneously in an effort to minimize classification error.

4.1.1 Adaptive Methods

There have been many attempts at creating adaptive activation functions in the past including adaptive cubic spline activation functions [87] as well as single function per layer optimization of rectified linear units [40]. Neither of these methods include more complex non-linear functions such as the sigmoid or hyperbolic tangent functions nor have either of these methods shown their effectiveness on modern datasets and neural network architectures. These pre-existing versions of adaptive functions have shown that there are clear advantages to providing adaptive capabilities for such functions such as reduced training times and increased accuracies. For these reasons, an adaptive model that can be easily integrated with existing techniques to improve overall accuracies of artificial neural networks has been explored to improve upon these concepts.

4.2 Adaptive Activation Functions

Optimization for each activation function was achieved by defining a convex cost function for the linear weighted combination of each activation function applied to an input. In order to better understand the operation that is being applied, this process can be visualized as a single entity in a neural network referred to as an "adaptive activation function layer" as shown in Figure 4.1. The individual activation functions are defined as f_i, where $i \in 1 \ldots N$. The function l refers to a continuous differentiable function which is applied to

each gate matrix g_i. This gate matrix represents a matrix of parameters to be optimized to find the best activation function by gating certain activation functions, while allowing others to express themselves. In the case of a convolutional neural network, a node would be representative of a single pixel from one layer to the next which essentially allows for each pixel to have a separate activation function which is dynamically optimized [22].

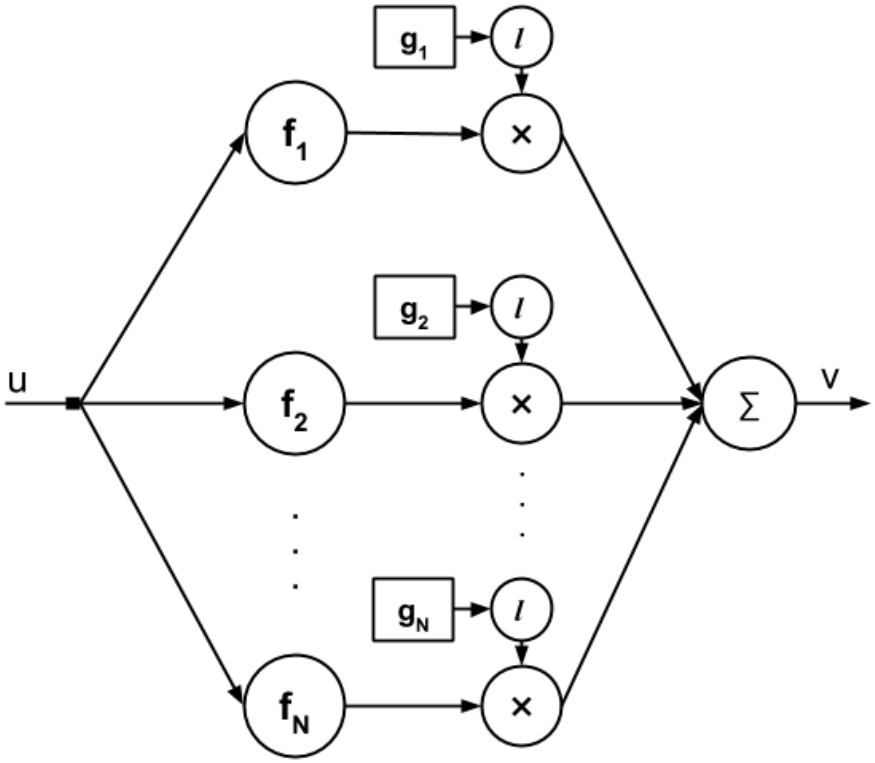


Figure 4.1: Block diagram of a single adaptive activation function layer [22].

Ideally, the values of $l(g_i)$ should fall within 0 to 1 which corresponds to having each activation function as either on or off respectively. These gate parameters are treated as optimization parameters in the gradient descent algorithm in order to allow for their optimal values to be solved for. The

output v is calculated by passing the input u through this layer as defined by (4.1) [22].

$$v = \sum_{i=1}^{N} f_i(u) l(g_i) \tag{4.1}$$

This adaptive layer essentially allows certain activation functions to express themselves more prominently, or less prominently based on their attribution to the overall computed cost of the layer. Ideally, the activation functions will either be fully blocked which corresponds to a 0 or fully expressed which corresponds to a 1. This would allow a more restricted, but more easily optimized subset of linear combinations of activation functions. In order to achieve this behavior, as well as to allow the gate parameters to be optimized, the gate limiting function, l, is defined as the sigmoid function in (4.2). This function was chosen due to its relatively simple to compute derivative as well as its nonlinear behavior which constrains the activation functions to be fully suppressed or fully excited. This also prevents the gate values from growing uncontrollably [22].

$$l(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

Given the definition of a multiple activation function gate layer, its corresponding gradient update equations were derived in (4.3) and (4.4). The parameters are as follows: δg_i represents the gradient update to the gate values, while δu represents the gradient update to the input to the adaptive

function layer [22].

$$\delta g_i = \delta v \frac{\partial l}{\partial g_i}(g_i) f_i(u) \tag{4.3}$$

$$\delta u = \sum_{i=1}^{N} \frac{\partial f_i}{\partial u}(u) l(g_i) \delta v \tag{4.4}$$

An additional cost term can be added to constrain the percentage of activation functions that are used in the network applied at every iteration. This could allow for more restricted behavior to be modeled that prevents the activation functions from becoming unstable or growing out of control.

The gradient update can also be scaled in order to prevent the adaptive functions from taking over the optimization problem. This issue essentially is due to the fact that there are many more variables added to the overall cost of the network which must be optimized. In CNN's where there may be more activation parameters than filter or fully connected weight parameters, the latter two parameters can easily fall into the local minima due to the emphasis of the adaptive function parameters. In order to avoid this, a scaling factor S_f can be introduced which scales the entire gradient update. Additionally, a momentum term is recommended to avoid local minima.

An alternative method to avoid this problem is to vary the scale term per epoch such that certain epochs only train the network's original parameters, while other epochs will include the adaptive activation function parameters. This can be done by setting the scale factor to 0 for a certain number of

epochs and then to any positive value for a specific number of epochs. Alternating in this fashion allows the adaptive activation functions to try to settle into the learned parameters of the network more naturally, however this needs to be explored further.

4.2.1 Activation Function Representation

An additional set of activation functions were created in order to show the effects of allowing more complex nonlinear functions to be included in the optimization process. Activation functions that are periodic in nature have been proposed which seek to introduce a more efficient nonlinear separation of data in earlier layers of a network.

The definition of such a function was based on a sum of sigmoid functions and its corresponding derivative. The overall equation allows for periodic behavior to occur centered around the origin with any given distance between humps, D and the specified number of half-humps, M. The following equations define this multimodal function generically using these two parameters to allow for more fine-tuned customization of the shape of the function.

$$J = ((M\%2) == 0)(i == \lceil M/2 \rceil)||(i \neq \lceil M/2 \rceil) \tag{4.5}$$

$$K = |J - 1| \tag{4.6}$$

$$R = \left(\left\lceil \frac{M}{2} \right\rceil \%2 == 0 \right) \tag{4.7}$$

$$offs = \left(\left(D \left\lfloor \frac{M+1}{4} \right\rfloor \right) - \left(\frac{D}{2} \right) (R) \right) - D(i-1) \tag{4.8}$$

$$y = \sum_{i=1}^{\lceil M/2 \rceil} \left(4J(1 - g(x + offs)) + K \right) g(x + offs) \tag{4.9}$$

(4.5) defines J as a binary function that determines whether or not there should be a full hump (1) or half-hump (0) in the summation for a specific iteration i, while (4.6) defines K as the logical inverse of this condition. An offset function is needed to keep the overall function centered around the origin which is given in (4.8). This is achieved by calculating the local center for each i^{th} equation as a function of the distance between functions, D and half-mode, M. Finally, the overall function is given in (4.9) wherein these variables are integrated together to form a summation of either a sigmoid or its derivative at each iteration with a specific offset. The $4J$ term comes from the fact that the derivative of the sigmoid function has a maximum of 0.25 which can be normalized to the asymptotic maximum of the sigmoid function by multiplying it by 4. This ensures that each function has relatively the same minimum and maximum values.

The derivative of the above defined multimodal function was also derived in order to allow for error to be calculated properly for any neural network using this particular activation function. Specifically the derivative of the

sigmoid function is given in (4.10) and the final derivative is shown in (4.11) which is just a summation of the derivatives of each individual function applied.

$$g'(x) = (1 - g(x))(g(x))) \tag{4.10}$$

$$dy = \sum_{i=1}^{\lceil M/2 \rceil} (4Jg'(x + offs)(1 - 2g(x + offs)) + Kg'(x + offs)) \tag{4.11}$$

A visualization of the multimodal function can be seen in Figure 4.2 which shows two separate cases of the function. The $M = 1$ case shown on the leftmost of the figure demonstrates that the sigmoid function is a subset of this function.

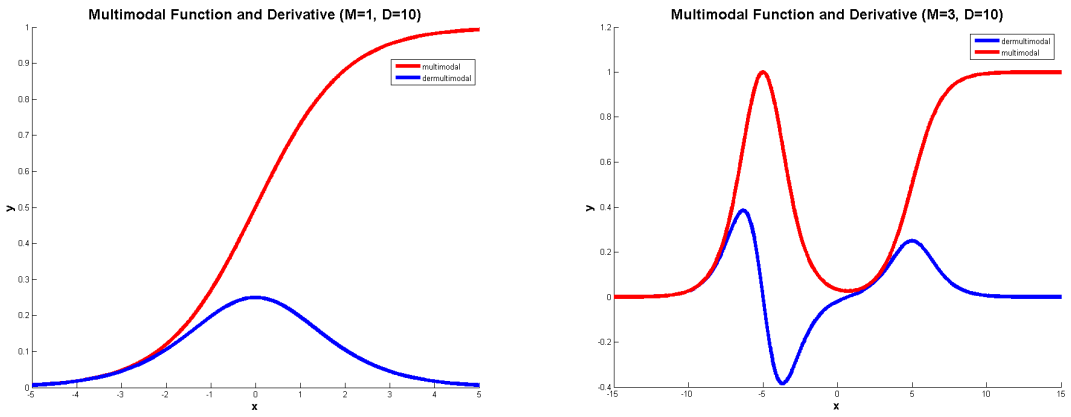


Figure 4.2: Examples of activation functions (in red), and their corresponding derivatives (in blue) used for backpropagation.

4.3 Results

Experiments are performed on the CIFAR100 and CalTech256 datasets. The CIFAR100 dataset has 100 classes with 500 images for training and 100 images for testing respectively per class and has an image size of $32 \times 32 \times 3$ pixels. The CalTech256 dataset has 257 classes, with 80 to 827 images per class with image size of $300 \times 200 \times 3$ pixels.

Deep convolutional neural networks are used to contrast using traditional activation functions (baseline) as compared to the adaptive activation functions introduced in this paper. In order to obtain a fair comparison between baseline results and adaptive function results, the same training parameters were used for both methods without scaling the learning rates while training. To determine the best way of training the adaptive function networks, two separate trials were conducted: 1) replacing baseline activation functions with adaptive activation function layers after the first few layers of a deep CNN, and 2) replacing baseline activation function layers with adaptive activation function layers at the last few layers of a deep CNN. The results from each experiment support recent findings in neuroscience as well as related deep learning research.

Experiments show that although the ReLU activation function works best when used in isolation, the traditional sigmoid and tanh occur more frequently when multiple activation functions are allowed to become active at each node. Networks which support multiple activation functions per node

are shown to significantly outperform networks with one activation function per node and also train significantly faster.

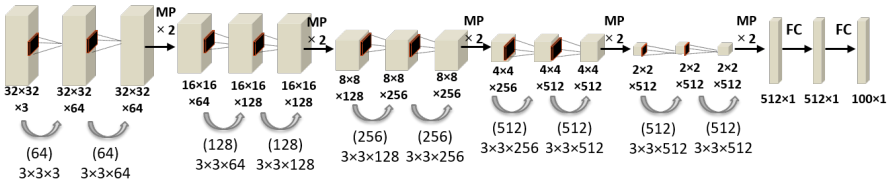


Figure 4.3: General convolutional network architecture for CIFAR 100 dataset [22].

4.3.1 CIFAR100

Baseline accuracies were obtained through the use of a VGG-like architecture [84] as shown in Figure 4.3 using batch normalization [49] to speed up the training process. The maximum testing accuracy obtained without random translations or flips was 57.5% after 300 training epochs as shown in Figure 4.4.

In order to determine the best place in the network to apply adaptive functions to, two separate experiments were run with adaptive activation functions using the *Sigmoid*, *Tanh*, and *ReLU* functions to adapt to the network. In the first network, the adaptive activation functions were applied to the first six layers where previously ReLUs were used. In this case, the overall training time of the network suffered and the testing accuracy dramatically dropped down to 51.3% at the end of 300 epochs. This is most likely due to the fact that the network was not able to generalize to the testing set with so many parameters to optimize in the first layers and therefore ended up doing poorly.

Next, the adaptive functions were applied to the last seven layers where previously rectified linear units were used. This configuration did much better than the previous and in fact produced a gain in accuracy of 2% when compared to the baseline case. Furthermore, the amount of epochs to reach the same accuracy for this adaptive case versus the baseline was less. These results can be seen in Figure 4.4 which compares the baseline training and testing accuracies to the adaptive case.

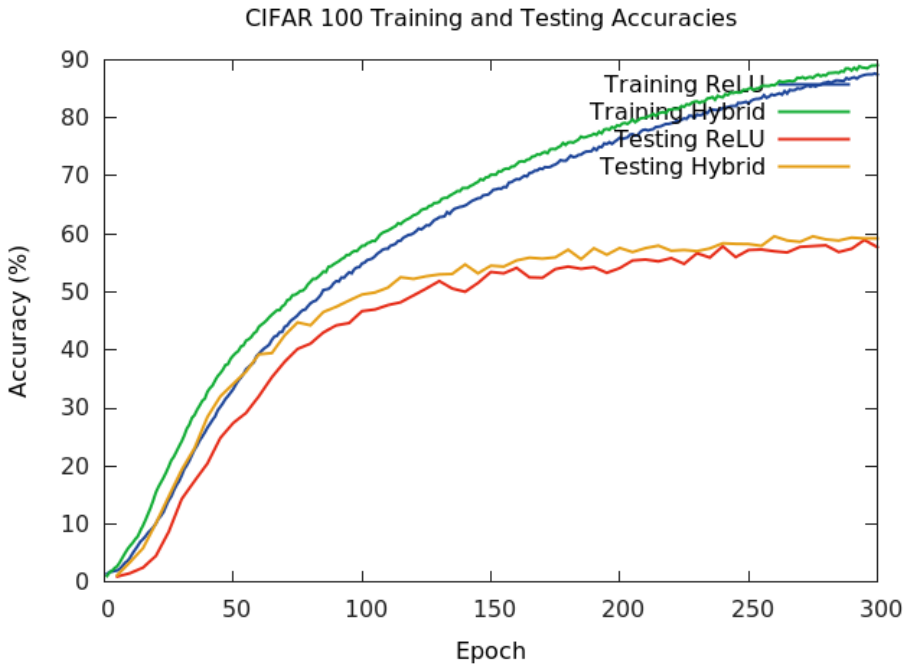


Figure 4.4: Accuracy comparison between baseline and adaptive functions for CIFAR 100 dataset [22].

From this comparison between the two methods, the adaptive functions seem to positively enhance the discriminative accuracy of a convolutional network when applied to the final layers of the network. It can be postulated that the early layers of the network are crucial in determining the overall

accuracy of the network and therefore if too many extra parameters are included, the overall number of epochs for the network to converge increases. The overall accuracy is also impacted by this since there are undoubtedly many local minima introduced by the adaptive activation functions which can halt the progress of the overall network if techniques to avoid them are not used properly.

The usage statistics of the adaptive networks are of importance to analyze since their layer-by-layer properties ultimately determine the success of the overall network. The total usage percentage of the final seven adaptive activation functions of the network are shown in Figure 4.5.

As shown by the layerwise usage statistics in Figure 4.5, the sigmoid activation function is used less than the other two functions in every case except for the third in which it ends up being used just as often as the hyperbolic tangent function. Surprisingly, the ReLU function is only dominantly used in the final activation function layer between the two fully connected layers of the network. From these statistics, one can determine that the tanh function when paired with the rectified linear unit function will be used most often in most layers, however the sigmoid function can also become useful in order to clamp the output of the overall activation function.

A random sampling of activation functions from a single layer are shown in Figure 4.6 wherein each adaptive function is independently optimized for a single node.

4.3.2 Caltech256

Similar methods were used to obtain accuracies for the Caltech256 dataset. Images were resized to a size of 64×64 which necessitated a change in the number of layers in Figure 4.3. Two convolution layers and a max pooling were inserted before the fully connected layers in order to reduce the dimensions of the image down to a single pixel by the end of the network. The decision to reduce the size of each image so drastically from their original sizes ultimately cost the network significant accuracy, however for the purposes of comparison with the adaptive case, this is not that important. The baseline accuracy of the model utilizing only rectified linear unit activation functions was 31.1% as shown in Figure 4.7. The placement of these adaptive functions ultimately decided the overall general improvement or degradation of accuracies in the overall model.

Two separate model strategies were adapted from the the CIFAR 100 experimental findings. The first strategy substitutes adaptive activation functions for the first four convolution layers, while the second strategy substitutes adaptive activation functions for the last five layers of the network. In the first case, both the training and testing accuracies were negatively impacted and after the same 300 training epochs, the final testing accuracy ended up at 28.3%. This mirrors the behavior of the CIFAR 100 experiments. This problem could most likely be avoided by scaling the gradient update to the adaptive functions, however this would result in comparatively

longer training times.

The second experiment shows a minor boost in both training and testing accuracies over the same number of epochs, however the overall testing accuracy was around the same as the baseline by the end of 300 epochs as shown in Figure 4.7. Both of the adaptive case curves are bowed outwards and to the left of the baseline case clearly indicating an advantage in training time reduction. Although the overall accuracies are the same after 300 epochs, this most likely is a limitation of the amount of data that is being processed in the network itself rather than a limitation of the adaptive functions. If the images passed into the network were considerably larger and the network architecture itself was deeper, the accuracies may improve further for the adaptive case for the same amount of time. This is due to the fact that the the network would generalize more readily to the test set if less information was thrown away from the original images. Larger images would mean that the testing accuracies would take longer to saturate and the boost in accuracies seen in the current model would be able to increase over a longer duration of time. It takes roughly 150 epochs for the current model's testing accuracies to saturate in both the baseline and adaptive case [22].

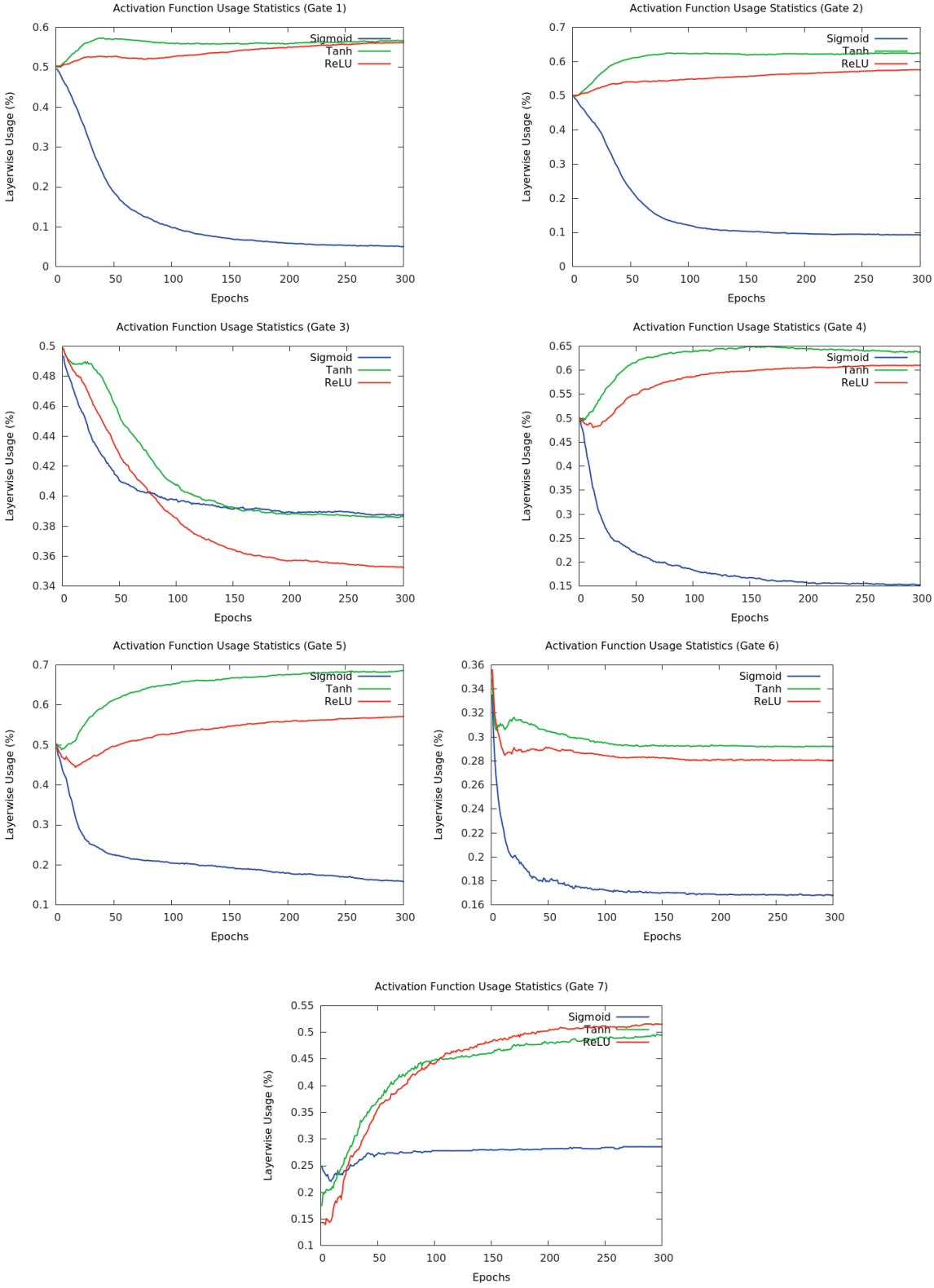


Figure 4.5: Activation function statistics per layer for the CIFAR 100 dataset [22].

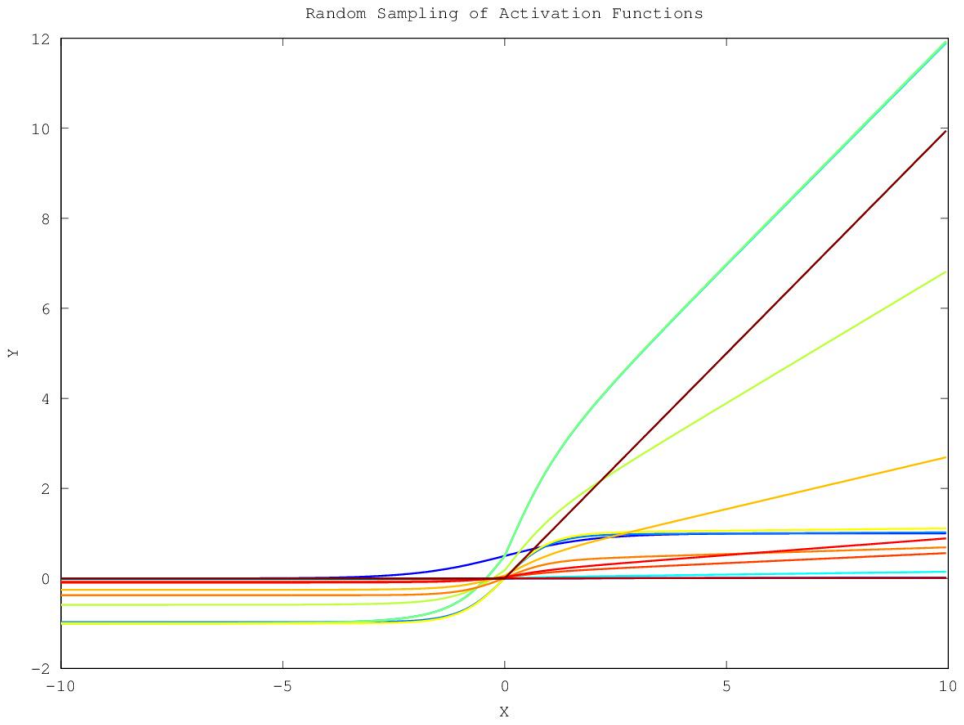


Figure 4.6: Random sampling of activation functions for CIFAR 100 dataset [22].

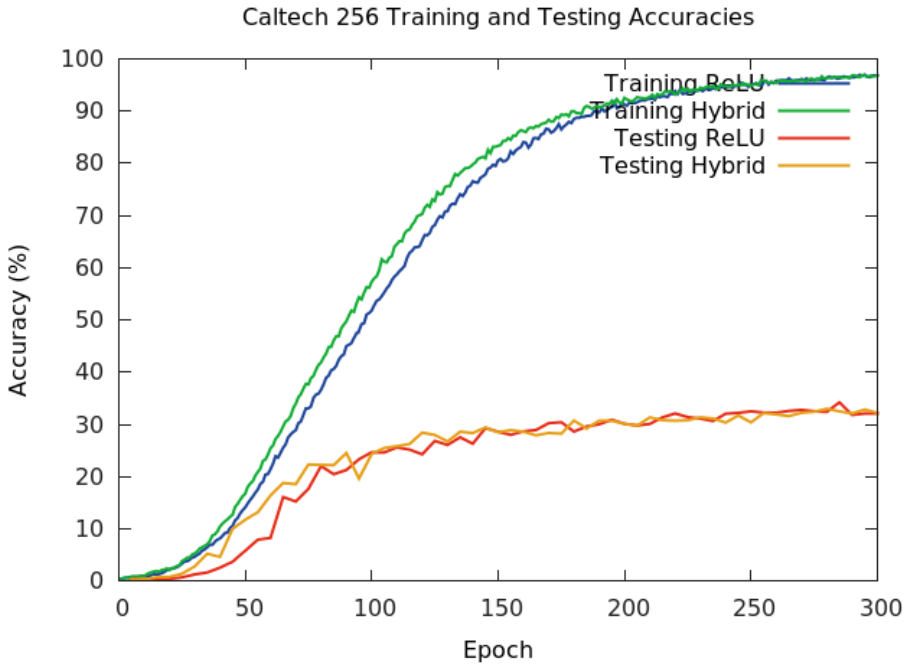


Figure 4.7: Accuracy comparison between baseline and adaptive functions for Caltech 256 dataset [22].

Chapter 5

Temporal Coherence

5.1 Rotoscoping

Rotoscoping is a traditional animation technique which maintains motion coherence between photorealistic frames and the resulting animated sequence. A rotoscope is an apparatus that projects an image onto a piece of glass (patented in 1917 by Max Fleischer), which was once used to manually trace edge information of the original frames. A diagram of the apparatus used in this process is shown in Figure 5.1. From these edges, the scene was then drawn, preserving the objects' shape and location through time. This allowed the coherence of the motion from the live action frames to be maintained while simultaneously rendering the artist's style onto the final sequence. Given the difficulty of producing non-mechanical human-like motion, rotoscoping was frequently used to obtain motion data for animated characters such that they mimicked lifelike motion. Further, the stable edges obtained by rotoscoping help reduce shaking and jitter in animation, making it more appealing to watch. Notable early animations utilizing rotoscoping include *Koko the Clown*, *Betty Boop*, *Popeye*, and *Snow White and the Seven*

Dwarfs [81].

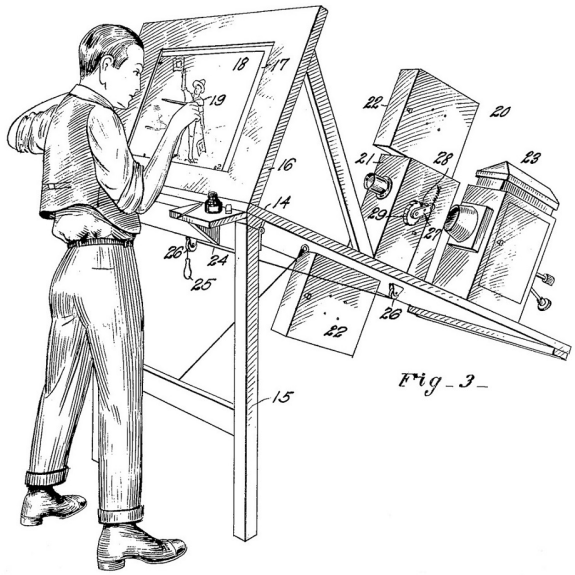


Figure 5.1: Patent drawing for Maxwell Fleischer's rotoscoping apparatus.

However, the merits of rotoscoping are not without cost. The act of physically tracing individual frames from a live-action video is an immensely time consuming process and requires highly skilled artists. Additionally, while the technique stabilized the animation, variation between tracings introduced jitter at the traced edges. With the rise of the personal computer, the rotoscoping technique could finally be refined to reduce these costs [81].

High performance computing stands to significantly reduce the time involved in rotoscoping, while the precise nature of computing means edges should be more accurately reproduced, reducing jitter. It is a facile approach to apply existing edge detection algorithms in order to obtain a rotoscoped result. Rotoscoping is an art form and as such, the decision of edges, their variation, thickness, style, and even transformation have traditionally

been selected by the rotoscoper - something not easily algorithmically reproduced. Further, the whole contents of each animated frame still need to be drawn, even after reproducing the desired edges [81]. Instead, various implementations help animators reduce the tedious work involved in rotoscoping. Crowdsourcing rotoscoping is an effective way of distributing the work involved [94] for individual images, but it is unlikely that many individuals would involve themselves in crowdsourced rotoscoping of numerous near-identical frames in videos. While the crowdsourced individuals could be compensated as motivation, the same compensation could be used for full time animators, with a vested interest in accuracy. Many software packages utilize interactive rotoscoping instead, which track edges through time (often via splines), allowing the animator to retain control over lines, edges, and their respective transformations.

As an alternative approach to these methods, one can consider to leverage optical flow data in order to track the motion of a scene over time in order to produce temporally coherent animations, however a naive approach using this method does not completely reduce jittering artifacts. Currently no completely automated rotoscoping method exists to allow an artist to both choose any style and apply it to an animation in a fashion that captures and reproduces the underlying motion of objects throughout time in a consistent and coherent manner. The following sections will discuss various methods of optical flow which were considered for the process that we propose as well as their strengths and weaknesses.

5.2 Optical Flow

Optical Flow refers to the apparent motion of objects in a scene caused by relative motion between the observer and the objects. This was originally described by a psychologist named James J. Gibson in the 1940s as a visual stimulus in animals moving through the world that allows them to perceive the shape, distance, and movement of objects in the world [32]. This visual perception affords the ability for these animals to make decisions about their environments based on the discerned motion of objects.

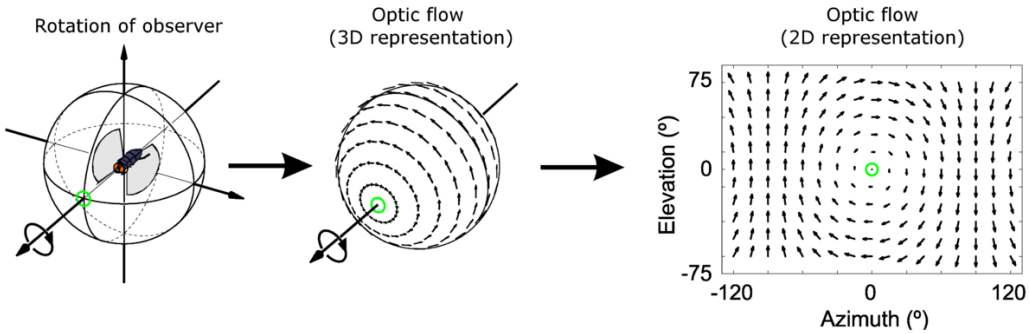


Figure 5.2: Relative optical flow field from rotating observer [48].

The use of optical flow data is crucial in the field of temporal computer vision allowing computerized systems to track and detect object movement. This data can further produce estimates of inherent 3D motion of objects relative to the observer, however the question of whether the objects are moving or whether the observer is moving is still unanswerable due to the fact that both contexts will produce the same optical flow field. Optical flow instrumentation in hardware systems allow image systems to detect motion such as within optical mice which use this mechanism for measuring the

movement of a mouse over a surface.

In 1981, Horn and Schunck defined a few key assumptions that would create a baseline for optimizing the problem of computing dense optical flow. These assumptions included brightness constancy and spatial smoothness. Limited by technology of the time, their model was somewhat dismissed as having unimpressive findings. Decades later, descendant methods augmented this original technique with improved optimization, robust error functions [7][9][59][77], median filtering [88], and course-to-fine estimation hierarchies [6] among others.

Brightness constancy describes the tendency for objects to maintain their relative brightness over time and is given by the constraint in (5.1).

$$I(x+u, y+v, t+1) = I(x, y, t) \tag{5.1}$$

The original image $I(x, y, t)$ thusly will almost exactly resemble the next frame unwarped by flow vectors u and v. This assumption is very basic and is not quite representative of all real life situations, however it can fairly generalize to most movements.

The second assumption of spatial smoothness refers to the fact that neighboring pixels are very likely to belong to the same surface although not always. This assumption is formulated in (5.2).

$$u_p = u_n, n \in G(p) \tag{5.2}$$

This assumption makes the spatial derivative equal to zero at these pixel locations wherein p refers to a center pixel, n refers to a neighbor pixel contained in the set of neighboring pixels $G(p)$. This assumption is also not complete since many times a center pixel may be on the edge of an object wherein some of its neighboring pixels are certainly not part of the same object nor do they have the same motion. The full objective function given these two assumptions is given in (5.3).

$$\begin{aligned} E(u,v) = &\sum_{s} \left(I(x_s + u_s, y_s + v_s, t+1) - I(x_s, y_s, t)\right)^2 \\ &+ \lambda \left(\sum_{n \in G(s)} (u_s - u_n)^2 + \sum_{n \in G(s)} (v_s - v_n)^2 \right) \end{aligned} \tag{5.3}$$

The quadratic penalty inherent in this objective function allows the error function to be convex and optimizable, however it also implies that the noise inherent in the solution is Gaussian. Over the years many methods have worked to improve upon these assumptions to make them more robust and less prone to the deficiencies discussed above.

The use of optical flow in motion estimation is integral in determining the temporal relationship between adjacent frames in a video sequence and is also currently used in object segmentation tasks for this reason. Among the methods to be discussed are Sun et al.'s non-local+ algorithm [86], EpicFlow [76], and PCAFlow [91]. These methods represent a wide variety

of optical flow methods however there are several more recent methods including those by Simonyan and Zisserman [82] who studied several variants of optical flow to dramatically improve semantic understanding of action in videos. This semantic understanding of videos with optical flow data can further augment the generative uses of optical flow data to produce more temporally coherent and recognizable motion.

5.2.1 Dense Non-Local Optical Flow

Sun et al. [86] thoroughly analyzed the assumptions and methods used in the Horn-Schunck method in an effort to systematically improve it, with some common augmentations to the algorithm. There were several different methods that were explored that adapted the baseline assumptions of Horn and Schunck including pre-processing, coarse-to-fine estimation, alternative interpolation methods, separate nonlinear penalty functions, and finally median filtering. Each of these "secrets" address specific deficiencies of the baseline method.

The pre-processing performed on the original images was shown to be useful, but simpler methods of derivative constancy filters were shown to be better than complex filters that decompose textures [86]. The coarse-to-fine estimation solves the problem of feature scales by utilizing a Gaussian pyramid with a specific downsampling factor that allows for downsampled versions of the image applied to the algorithm to have their results affect the overall optimization of the flow. This singular change managed to achieve

significant improvements on modern datasets when combined with a non-convex generalized Charbonnier penalty function and spline-based bicubic interpolation. Recognizing the benefit of median-filtering during optimization, Sun et al. [86] refined the objective function to incorporate a median filter in the form of extended neighborhoods, however they modified this to minimize negative impacts of median filtering, such as over-smoothing [86]. The result was a method that weighted the impact of the median filter with non-local insights wherein the weights were learned to represent how likely a pixel is to belong to a center pixel's surface.

All of these modifications were applied independently at first and then merged together in order to form a much more robust method referred to as **Classic+NL-Full**, however a reduced computation version called simply **Classic+NL** was shown to be able to reach substantially the same accuracies in a fraction of the time. Both of these methods were the highest ranking algorithms of their time on the Middlebury optical flow dataset [3] and still perform fairly decently when compared to more modern methods. The improvements gleaned from the systematic and incremental improvements that were explored in this paper had quite a large impact on future optical flow papers throughout the years following it.

5.2.2 EpicFlow

Another separate more recent algorithm for computing dense optical flow is referred to as "EpicFlow" or Edge-Preserving Interpolation of Correspondences for Optical Flow. This method takes into account the fact that coarse-to-fine estimation causes error to be propagated from the coarsest layers to the finer layers causing much of the error in preserving edges of objects [76].

In order to fix much of these problems, an optical flow method was produced that integrates a dense correspondence field detection algorithm using deep descriptor matching [89]. This generates a set of matching points between two frames in order to traditionally track objects, however its inclusion in initializing the optical flow field significantly reduces the tendency for the optical flow algorithm to fall into bad local minima. This is largely due to the fact that these descriptor points largely impact the details that are preserved between frames and help the motion estimation immensely.

Secondly the algorithm integrates contour matching in order to detect motion boundaries such that natural boundaries will form preventing localized clusters of velocity vectors from overstepping object edges. This is achieved using a recent state-of-the-art edge detector called Structured Edge Detector [19]. The combination of both of these extraneous bits of data allow EpicFlow to perform very well on modern datasets such as MPI-Sintel

[11] and Kitti [31] while also reducing the computation time needed to optimize and compute flow matrices [76]. The overall computational pipeline for this algorithm is shown in Figure 5.3.

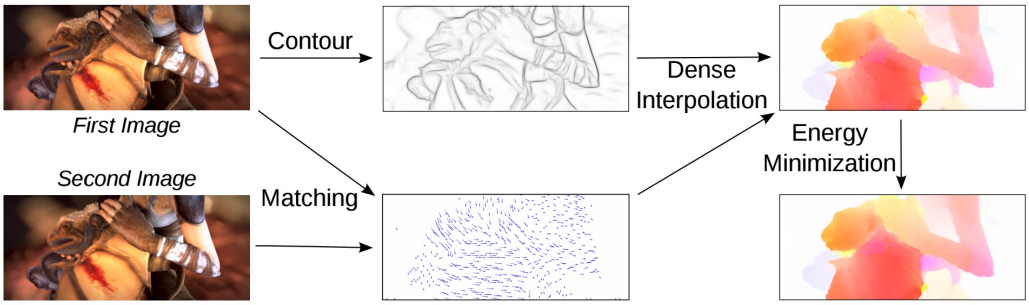


Figure 5.3: Overview of the EpicFlow computational pipeline [76].

Much of the failure cases in EpicFlow are due to missing contours and mis-matched descriptor points which can cause significant loss in detail in the output flow field. This occurs very prominently in thin structures [76]. Such a deficiency of the overall algorithm can be overcome by advances in descriptor matching algorithms, however it remains as a very real drawback in utilizing this algorithm in real-world optical flow detection.

5.2.3 PCAFlow

A very recent method for optical flow which significantly reduces the algorithmic and computational complexities of this calculation is called PCAFlow [91]. Although a regression to past technologies, the PCAFlow algorithm focuses on formulating a faster method of computing optical flow that doesn't sacrifice too much accuracy through the use of Principle Component Analysis (PCA) to form a basis for computing flow vectors very quickly. This

is integral to its application to real time devices such as cameras which can benefit from such fast dense optical flow estimation.

A comparison of the various methods discussed is shown in Figure 5.4 wherein the runtime versus average error of various dense optical flow methods is shown. These methods were all evaluated on the Sintel and KITTI datasets and the "Ours" labeling refers to PCA-Flow and PCA-Layers, both of which are CPU-based algorithms.

A careful attention to the tradeoffs of accuracy versus computation time must be taken into account when a specific algorithm is to be chosen for a task involving optical flow computation. These tradeoffs will be carefully analyzed in the following methodology chapter.

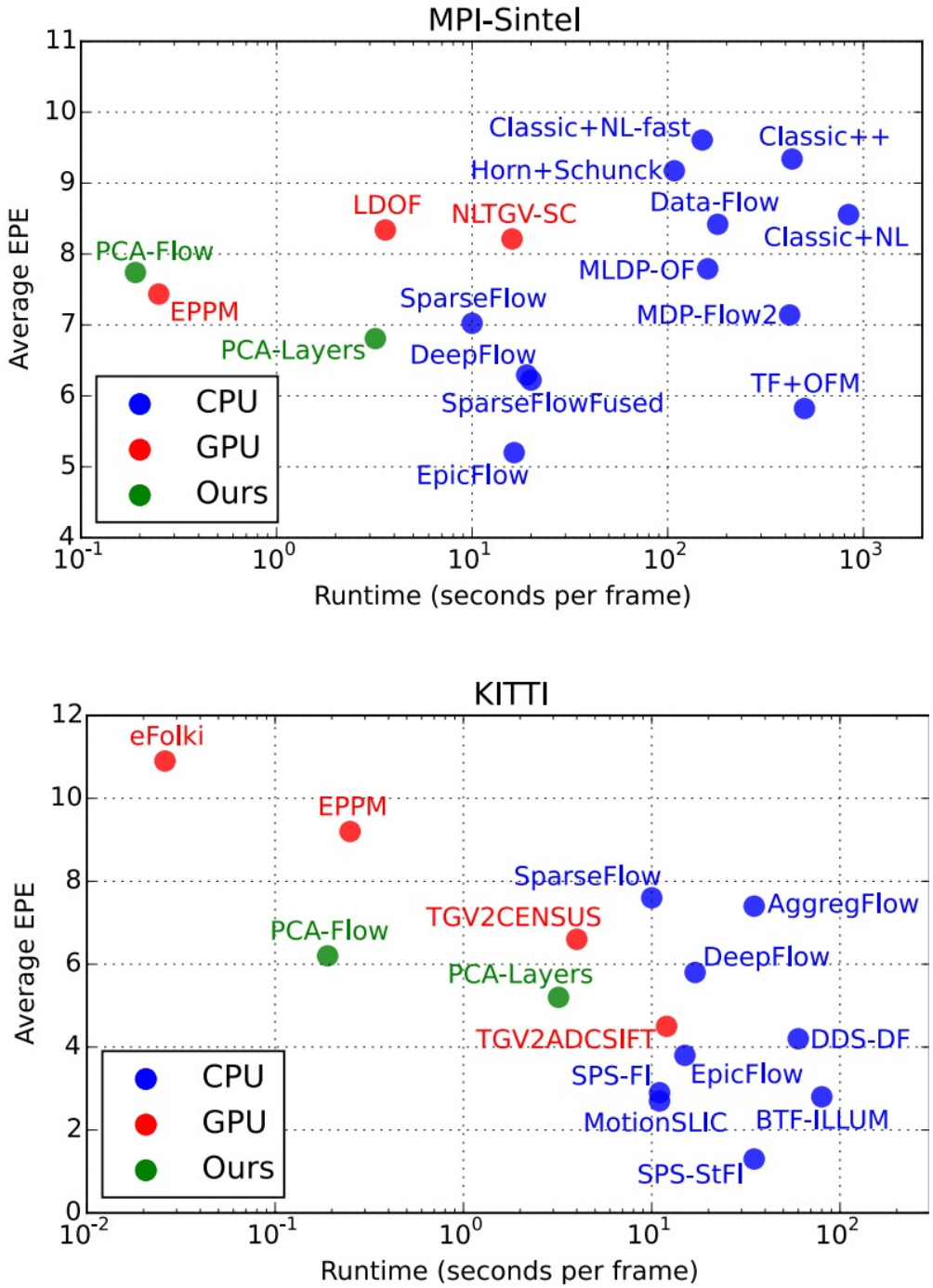


Figure 5.4: Average Endpoint Error (EPE) vs. runtime of various optical flow methods [91].

Chapter 6

Methodology

There are several factors to consider when producing an effective motion-tracking stylization technique for video sequences. One important aspect is the reduction of unwanted jitter, or noise in the motion of style information between frames. This can become very distracting to a viewer and often disorienting if there is a large amount of movement between frames [26][55]. Another aspect of temporal coherence comes from matching the transfer of style between two neighboring, but unrelated frames such as a jump cut within a video. Finally, realistically stylized animation will also consider overall frame brightness and contrast, whereby overexposed, underexposed, and low contrast frames warrant less style than mid-tone, high contrast, and colorful frames. All together, these three aspects create a more complete and attractive visual experience.

6.1 Stylization

A VGG-style convolutional neural network (the same one utilized in Chapter 4) which was pre-trained on the ImageNet dataset was utilized for the stylization of frames in order to leverage its hierarchical decomposition of

images into their component features including textures, edges, and objects. In this network, style loss and content loss are computed at each layer as described in [29]. This was more specifically achieved by defining two separate loss functions, one for content preservation and one for style preservation.

The goal of content transfer is to maximally represent the overall objects and details of a scene that make it recognizable to a human observing it. In the hierarchy of a CNN, the layers that are responsible for preserving this content are towards the end of the network and are able to recognize and classify individual objects. It is thus possible to preserve the output of a CNN layer of an image $\hat{x}$ by first forwarding the original image through the network and then saving the feature maps $\hat{F}$ at the layers of the network that are responsible for object-level details.

$$\begin{aligned} L_{content}(\hat{x}, x, l) =& \frac{1}{2} \sum_{ijk} \left(F^l_{ijk} - \hat{F}^l_{ijk} \right)^2, \\ & F^l \in \mathbb{R}_{N^l \times H^l \times W^l} \end{aligned} \tag{6.1}$$

$$\frac{\partial L_{content}}{\partial F^l_{ijk}} = \begin{cases} (F^l - \hat{F}^l)_{ijk}, & F^l_{ijk} > 0 \\ 0, & F^l_{ijk} \leq 0 \end{cases} \tag{6.2}$$

The current N feature maps of dimension $H \times W$ at all layers $l \in l_{content}$ in the network given by F^l are continually optimized with the convex loss function in (6.1) to match the ground truth values in order to maximally

represent the content at the given layers. The corresponding derivative of this loss function is shown in (6.2).

The objective of preserving the style of an arbitrary artwork $\hat{s}$ is quite a bit more of a complex problem. The overall style of a piece of artwork should be composed of the hues that make up the artwork as well as localized textures and even individualized structures. These lower level aspects of an image are categorized in the lower level of the feature hierarchy pertaining to textures and edges. These early layers of the CNN should maximally represent the textures of a given artwork, but not copy the original textures. In order to make the original textures of an example style image both rotation and shift invariant and allow for many possible solutions of the stylization to be produced, the Gram Matrix of the feature maps at specific layers are utilized instead of the activations themselves. The Gram Matrix introduces a surface of linearly combined entries wherein many possible solutions to the loss function can be found. The Gram Matrix is simply defined as a matrix multiplication between an original feature matrix F and its transpose F^T as shown in (6.3).

$$G = FF^T \tag{6.3}$$

The ground truth style Gram Matrices are produced by forwarding the original style image $\hat{s}$ through the network and then computing the Gram Matrix of the feature maps F^l at all layers $l \in l_{style}$. The corresponding loss function and its derivative are given in (6.4) and 6.5 respectively.

$$L_{style}(\hat{x}, x, \hat{s}, l) = \frac{1}{4(N^l H^l H^l)^2} \sum_{ijk} \left(G^l_{ijk} - \hat{G}^l_{ijk} \right)^2, \quad G^l \in \mathbb{R}_{N^l \times H^l \times H^l} \tag{6.4}$$

$$\frac{\partial L_{style}}{\partial F^l_{ijk}} = \begin{cases} \frac{(F^l)^T}{(N^l H^l H^l)^2} \left(G^l - \hat{G}^l \right)_{ijk}, & F^l_{ijk} > 0 \\ 0, & F^l_{ijk} \leq 0 \end{cases} \tag{6.5}$$

The combination of both of these loss functions into the final optimizable loss function is given in (6.6).

$$L = \sum_{l \in l_{content}} \left(w^l L_{content}(\hat{x}, x, l) \right) + \sum_{l \in l_{style}} \left(w^l L_{style}(\hat{x}, x, \hat{s}, l) \right) \tag{6.6}$$

The specific choice of layers for the lists $l_{content}$ and l_{style} were chosen with regard to the recommendations in [29] due to the specific hierarchical abstractions inherent within layers of the VGG network utilized. The choice of layers is shown graphically below in Figure 6.1.

The individual convolutional layers all have ReLUs applied after them and are represented as three-dimensional blocks of filters wherein a reduction in size indicates a max pooling operation.

6.1.1 Fine Tuning

A Torch7-based [14] implementation of the process described above was produced and released by [50] which allowed various stylization strategies

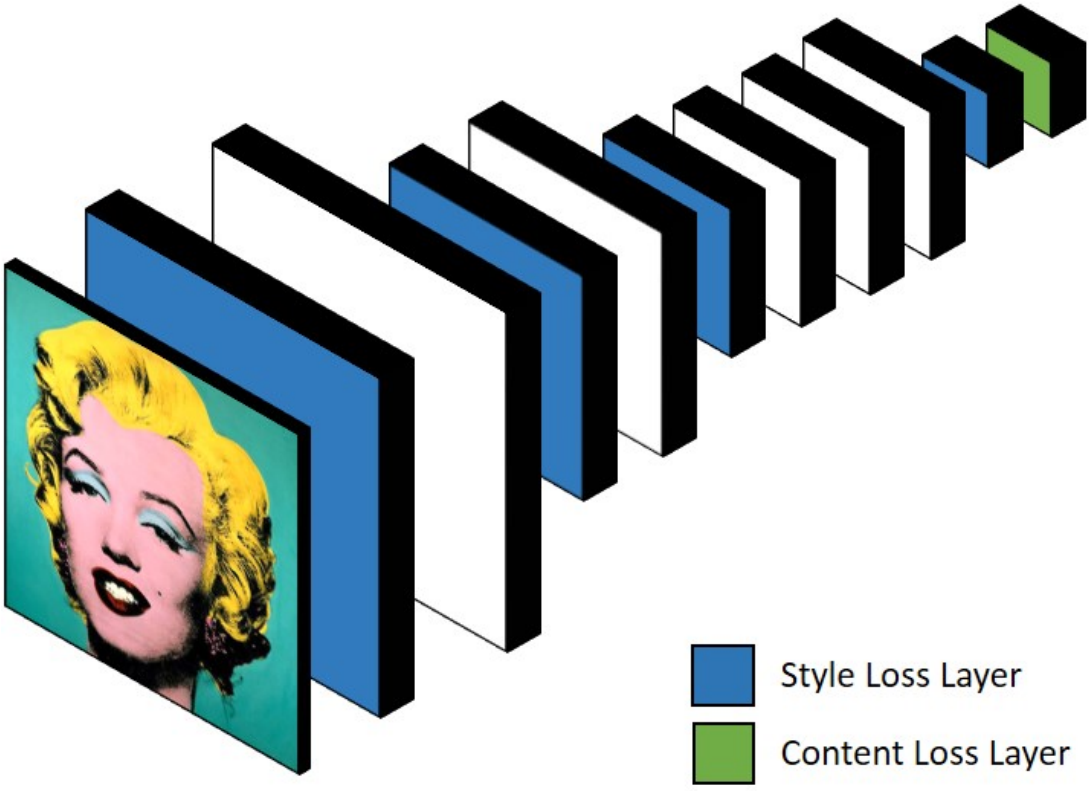


Figure 6.1: Layerwise loss choices for VGG-style network.

to be tested very rapidly.

In order to finetune the process described in [29] to produce consistently aesthetically pleasing quality images given any input style, many experiments were undergone utilizing various hyper parameters that are used to tweak the underlying weight of content transfer and style transfer as well as the optimization algorithm itself.

We discovered that the use of the Adam [53] optimization strategy with specific rules was equal in the quality of results produced by the LBFGS algorithm [41][4], however the execution time and memory requirements of the former method were far less than the latter. The reason that LBFGS

utilizes more memory is due to the fact that the algorithm stores a limited number of previous positions and gradients to implicitly perform operations that require the inverse Hessian [41]. Adam on the other hand does not store these large dense matrices and instead computes bias-corrected first order and second order moment vectors based on only the current gradient [53]. It is described that LBFGS performs faster on smaller datasets in particular [41][4] which is most likely the reason that it takes longer to converge on our problem since there are a large number of parameters to optimize. Our method utilizing Adam effectively reduced the overall number of iterations required to converge by over 10% and the overall memory required by around 20% allowing larger and higher quality pictures to be produced by the same GPU device. The maximum output size is also affected by the size of the style image itself since the activations are stored in the network on the GPU itself. Our experiments were all performed on a NVIDIA Titan X GPU.

The guidelines and modifications that we have determined produce consistent results are given below:

- Normalize the gradient calculations at each loss layer. This produces more coherent textures for Adam.

- Initialize the output image with the content image or a combination of mostly the original content image and a small amount of noise. This is in stark contrast to [29] which only reports results of outputs initialized

with random noise.

- Subsequently weight the style loss at least 2 orders of magnitude higher than the content loss. This is due to the fact that the content is mostly preserved from initialization and the style is more important to generate.

- Optionally desaturate the initialization images to allow the color palette of the style image to dominate when stylized.

- Optionally resize the style image such that its feature sizes with regards to the content image are the close to the size that the artist desires them to be on the final image.

With this set of heuristics, the resulting image was stylized using *The Starry Night*, by Vincent Van Gogh as the style image as shown in Figure 6.2.

An additional example of this stylization process is shown in Figure 6.3 which shows the breadth of applicability that these criteria are on any possible style and content image imaginable.

(a)

(b)

(c)

Figure 6.2: Example stylization of *Maryln Monroe* by Andy Warhol (b) in the style of Vincent Van Gogh's *The Starry Night* (a). The result of this fusion is shown in (c).

(a)

(b)

(c)

Figure 6.3: Example stylization of the author (b) in the style of Andy Warhol's *James Dean* (a). The result of this fusion is shown in (c).

6.2 Temporal Coherence

6.2.1 Optical Flow

In order to address the temporal coherence between frames, dense optical flow data is used to "seed" the initialization of the next frame's style information. In this way, the next frame will most likely fall into a minima that more closely resembles the previous frame with respect to its motion. This can be accomplished through the extraction of the previous frame's style using (6.7). This equation shows how the style is removed from the output image wherein S^t refers to the extracted style for frame t, the term V^t corresponds to the final generated frame at time t, and U^t refers to the original frame at the given time.

$$S^t = V^t - U^t \tag{6.7}$$

After style extraction, a warping operation (6.8) is applied to every pixel in the style in order to allow its information to be passed onto the next iteration. The updated pixel locations are based on (6.9) wherein v_x and v_y correspond to the computed optical flow velocities of each pixel at their respective x and y coordinates in the image. This essentially preserves the information about the previous style that was applied at the updated pixel locations where objects and larger structures have moved in order to allow the localized style to follow the motion of the underlying content.

$$f : U_{xy} \rightarrow U^*_{x^*y^*} \tag{6.8}$$

$$< x^*, y^* >=< x - v_x, y - v_y > \tag{6.9}$$

The resulting operation becomes (6.10) which computes the initialization for the frame $G^t_{x,y}$ based on a scalar multiple k of the extracted style from the previous frame warped using a linear transformation of the pixels which is then added to the original frame for time t.

$$G^t_{xy} = U^t_{xy} + kf(S^{t-1}_{xy}) \tag{6.10}$$

The processing pipeline of this operation can be seen in Fig. 6.4 wherein the extracted style is enhanced for detail.

The initialization of the stylization process with $G^t_{x,y}$ reduces the total number of iterations required to converge based on the chosen value for the scaling factor k.

A Torch7-based [14] program was developed which implements this stylization and warping process in order to allow for videos to be rapidly stylized with many tuneable parameters.

In our experiments we were able to reduce the total number of iterations T required to converge to acceptable outputs by 50% with a k value of 0.5, effectively speeding up the process by almost 2 times when compared to a fully sequentially processed baseline. This speedup, however does not

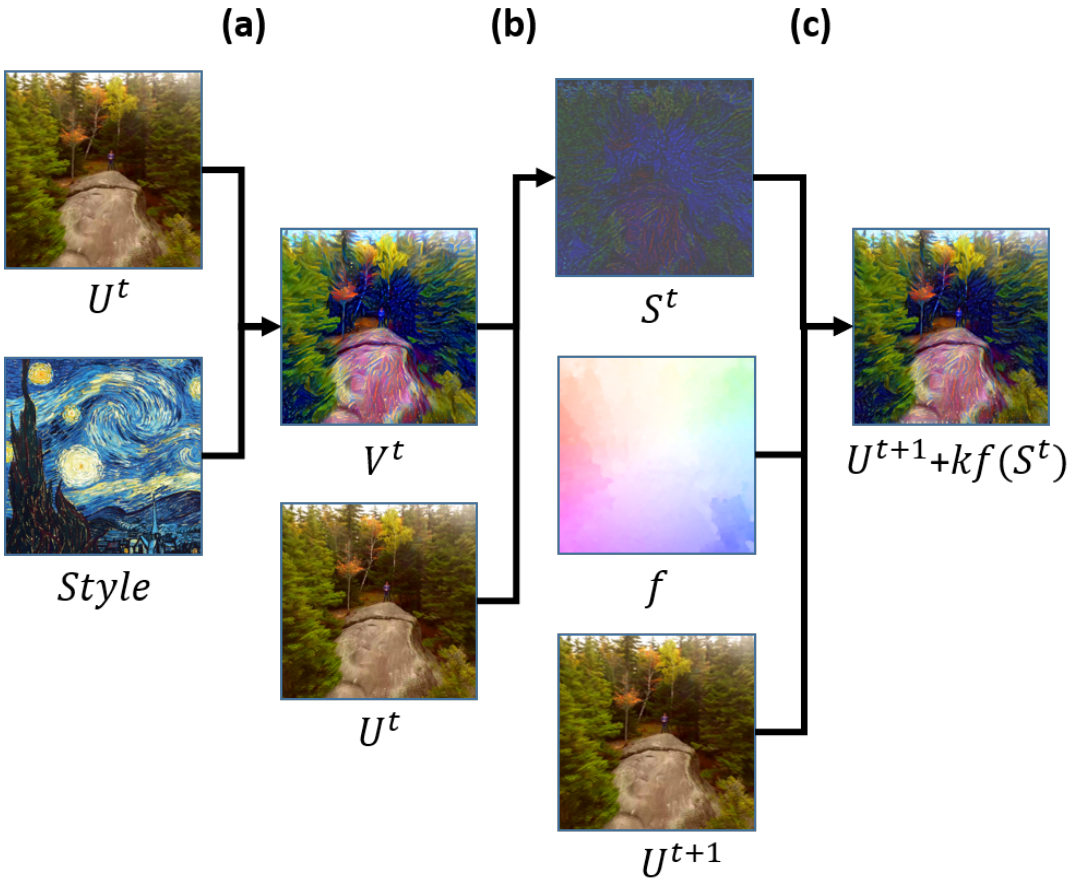
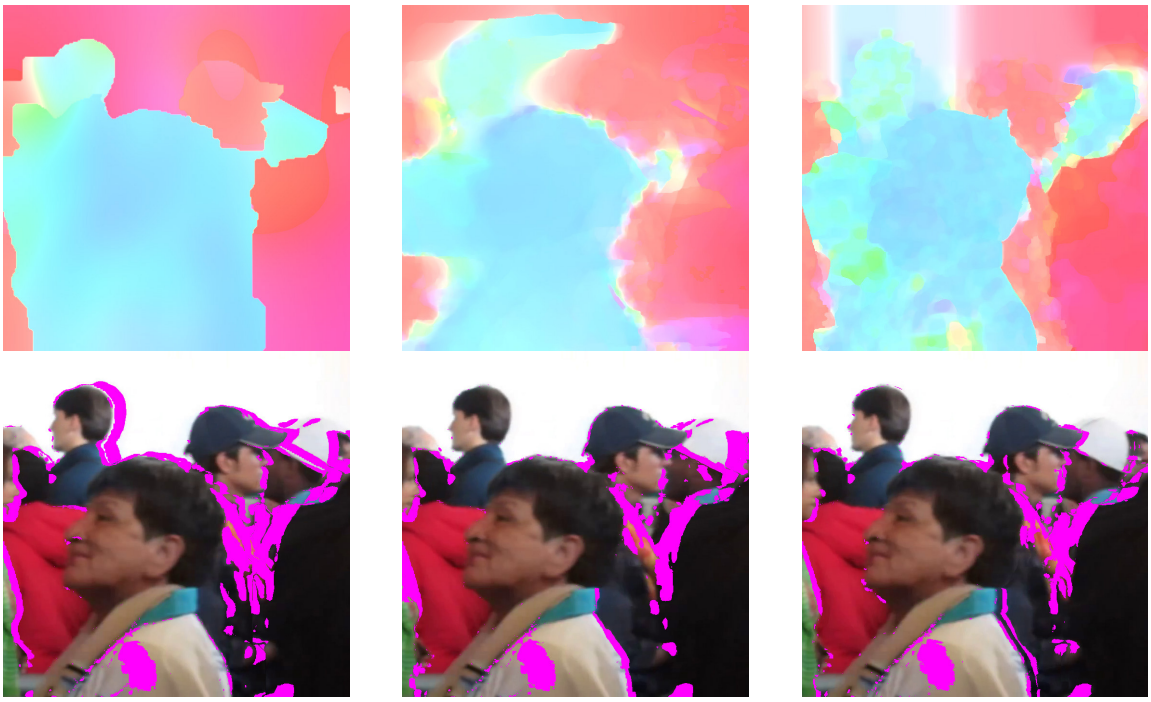


Figure 6.4: Processing pipeline of operations. (a) Stylization step using "This is Vermont Foliage" as the content input and "The Starry Night" as the style input. (b) Style extraction step. (c) Next frame seeded style with optical flow warping.

quite scale linearly and can be highly variable based on the optimization method used as well as the optimization parameters themselves. Further, the optical flow method has data dependencies that are not present in the naïve implementation making it considerably less parallelizable.

We compared the various optical flow methods mentioned in the previous chapter in order to determine which algorithm to utilize to produce our results. In order to compare these methods, we chose the same video *People*



(a) (b) (c)

Figure 6.5: Flow vectors (top) flow errors (bottom). Errors are shown in magenta. From left to right: PCAFlow (a), EpicFlow (b), Classic+NL (c).

Walking Around [10] for the optical flow to be computed on.

The corresponding flow vectors and error maps for each method are shown in Figure 6.5 for a selected frame wherein magenta regions denote regions where the warped original image does not match up with the next time step.

The overall execution time was taken as an average over all frames for each algorithm and the error was estimated as the normalized squared difference between the next frame $I(x, y, t + 1)$ and the warp of the previous frame for each computed flow vector $I(x + u, y + v, t + 1)$ as shown in (6.11).

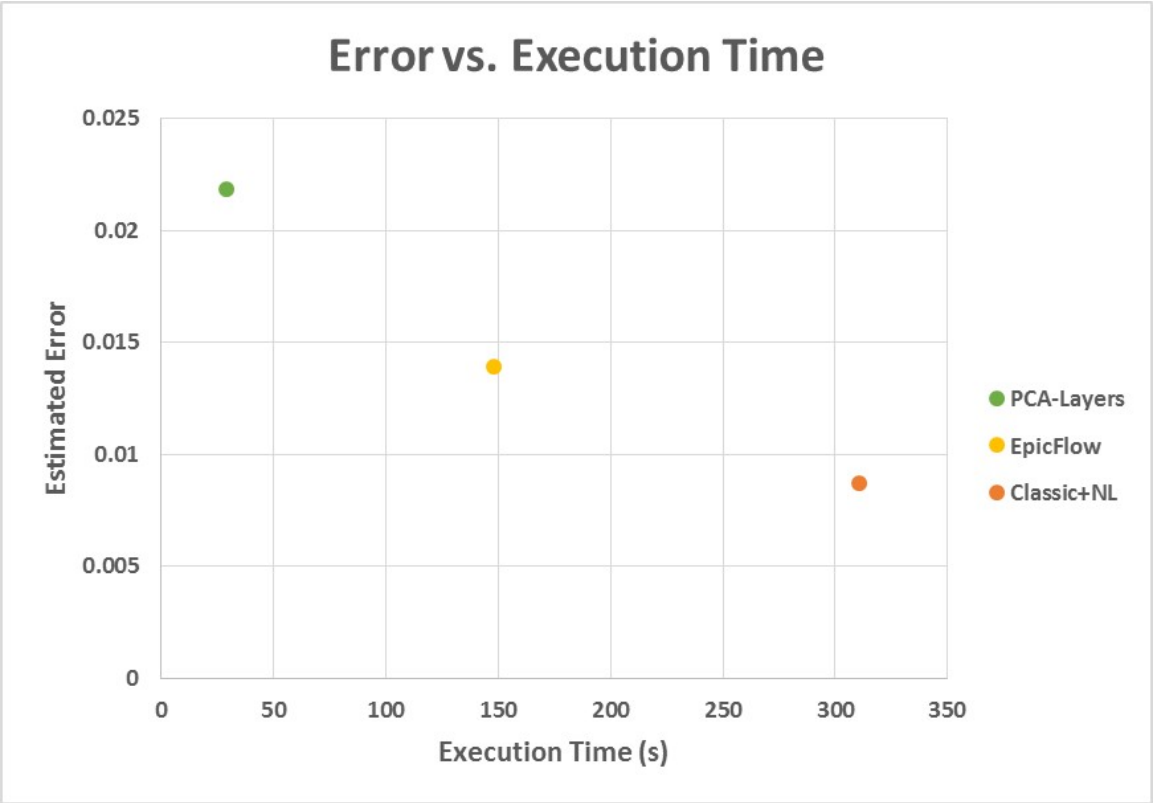


Figure 6.6: Comparison of different flow algorithms.

$$E = \frac{1}{N} \sum_{ijk} \left(I(x, y, t+1) - I(x+u, y+v, t) \right)^2 \tag{6.11}$$

From this, we produced a graph of the average error among frames for each method versus the execution time shown in Figure 6.6. All results are with respect to CPU implementations of each algorithm run on an AMD A10-7850K.

The error regions are very similar in the EpicFlow and Classic+NL cases, however small background details are missing in the EpicFlow output. The

PCA-Layers approach very much sacrificed accuracy for a faster execution time making it somewhat undesirable for stylization. After considering these results, we decided to utilize the Classic+NL approach to maximize the flow accuracy. What is also interesting to note is that although Classic+NL does not perform as well on the MPI-Sintel dataset, it generalizes slightly better to our real-world videos.

6.2.2 Adaptive Methods

The above method only handles a subset of motion cases wherein the style is applied uniformly with the same number of optimization iterations for every consecutive frame after the initial frame. This does not, however take into account the fact that in real video, there are discontinuities between frames called "jump-cuts" and also transitions called "fades" which can gradually decrease the contrast of a scene from a selected color. Fades may also be present as a natural alteration in the brightness within a scene over time which should be accounted for within the stylized output. We developed a more robust method that addresses these aspects via three separate adaptations: adaptive style based on motion acceleration, mean frame brightness, and RMS frame contrast.

The acceleration of the apparent motion of pixels can be thought of as the second derivative of the position of these pixels over time which is also equivalent to the first derivative of the optical flow velocities for each pixel. This acceleration is calculated in (6.12) wherein v^t represents the optical

flow matrix at time t.

$$\frac{\partial v}{\partial t} = \frac{v^t - v^{t-1}}{(t) - (t-1)} = v^t - v^{t-1} \tag{6.12}$$

The mean acceleration (6.13) is used to compare to a threshold value μ_{th}, which may be set in order to allow stylization to go unchanged if the acceleration is low and gradually reduce the transfer of style between frames using a new k_{mod} value for the style transfer weighting as the acceleration goes above the threshold. Simultaneously, the value of T is increased to a maximum T_{max} as the acceleration increases in order to apply a proportional amount of stylization to the new frame. The steepness of these thresholded Gaussian curves can be controlled using w_k and w_T.

$$\bar{\mu} = \frac{1}{NM} \sum_{ij} \left| \frac{\partial v}{\partial t} \right|_{ij} \tag{6.13}$$

$$k_{mod} = \begin{cases} k e^{-\frac{(\bar{\mu} - \mu_{th})^2}{2w_k^2}} & \bar{\mu} \geq \mu_{th} \\ k & otherwise \end{cases} \tag{6.14}$$

$$T_{mod} = \begin{cases} \left\lceil T + (T_{max} - T) \left(1 - e^{-\frac{(\bar{\mu} - \mu_{th})^2}{2w_T^2}} \right) \right\rceil & \bar{\mu} \geq \mu_{th} \\ T & otherwise \end{cases} \tag{6.15}$$

The Gaussian shape for these equations was used in order to allow slow onset of jump-cut transitions to adapt the style transfer, however as the jump-cut takes less time, the style transfer becomes dramatically less between frames. The above metric can have less of an effect for videos wherein

the inter-scene difference causes the apparent acceleration to be similar. In these cases it is more advisable to replace the second derivative of the optical flow data in (6.13) with the second derivative of the inter-pixel differences. This can be easily computed and has been more effective in practice. We found that although this acceleration adaptation can smoothen out transitions, it is not entirely necessary since adjacent frames with very different content will naturally have optical flow data that scatters their initializations. This method does, however help to increase the amount of style that is applied during sudden transitions.

The final adpative method used to improve stylization was an approach that analyzed the brightness and contrast of the image to adapt the style output of the image to changes in global lighting conditions. We first compute the overall RMS contrast of the image using (6.16).

$$C = \sqrt{\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(U_{ij} - \bar{U}\right)} \tag{6.16}$$

If this global RMS contrast is below a threshold C_{th}, but not below an absolute minimum threshold T_{min}, then the contrast is stretched linearly from the minimum brightness, 0 to the maximum brightness of the image, 1 based on (6.17).

$$U^* = (U - U_{min})\left(\frac{C^+ - C^-}{U_{max} - U_{min}}\right) + C^- \tag{6.17}$$

The above method stretches the histogram of brightness values of an

image past the threshold RMS contrast where C^+ and C^- are chosen to push the contrast to be relatively equal to the threshold. Once this threshold is reached, the histogram stretched frame is then stylized using an appropriate amount of transfer from the previous frame. If however the T_{min} threshold is passed, the frame is considered a solid frame and no style is applied to the current frame nor is any style transferred forward to the next frame. Once the frame has an appropriate amount of style applied, the histogram is stretched back to match the original contrast of the image.

Next the brightness is adapted using (6.18) wherein if the mean brightness, $\bar{\beta}$ exceeds a threshold β_{th}, then the overall brightness is reduced back to this threshold.

$$U^* = U\left(\beta_{th}/\bar{\beta}\right) \tag{6.18}$$

This reduces the overall brightness such that when the frame is stylized, the output image more closely resembles the input frames rather than oversaturating them. After the style has been applied, the original brightness of the image is restored. These two adaptations work surprisingly well across a wide range of lighting conditions and prevent unwanted streaking artifacts that can appear from optical flow data interacting with solid colored frames.

Chapter 7

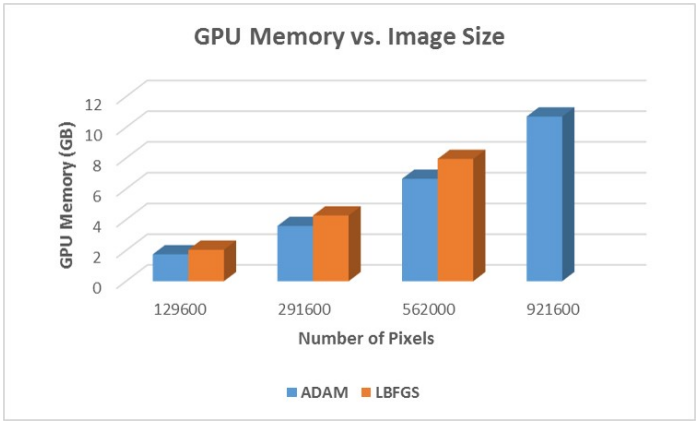
Results

In order to experimentally verify our methods, we utilize videos with both dynamic scenes and varied amounts of motion. To demonstrate the effectiveness of our method, the videos “This is Vermont Foliage” by Matt Benedetto [5] and “People Walking Around” by ButForTheSky [10] are used in this paper. The former is a video that shows very smooth continuous motion of the frame itself as the camera moves outwards through the scene while the latter video was selected for its complex movement of people with different speeds of walking and occlusions. These videos exemplify the wide range of movement that can be present in any video. The reason that traditional optical flow video datasets were not utilized is to show how well our video stylization algorithm works in situations that are generalized to the real world. Such datasets that contain ground truth optical flow data are not quite as interesting to evaluate from a non-empirical standpoint and the results that we obtain demonstrate the breadth of the practical applicability of our algorithm.

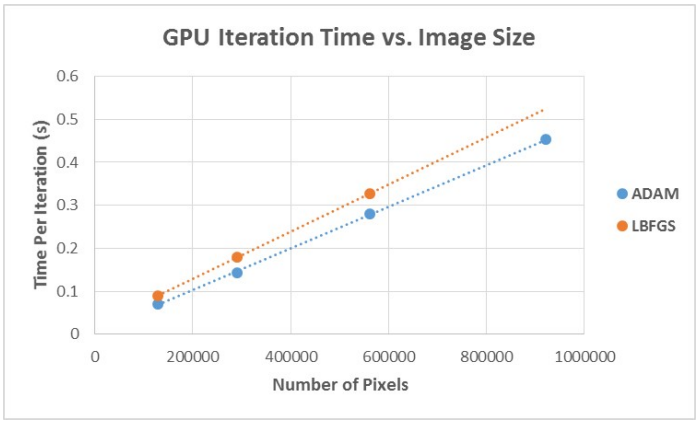
We profiled the GPU performance of two separate methods in order to

determine the overall memory and timing benefits that our adaptations provide. The baseline non-adapted noise-initalized case as described in [29] used the Limited Broyden-Fletcher-Goldfarb-Shanno (LBFGS) optimization algorithm [41], while our adapted heuristic version utilizes the Adam optimization algorithm [53]. Both methods were tested on the video "People Walking Around" [10] with individual frame sizes of 480×270, 720×405, 1000×562, and 1280×720, the final of which would be the best expected output resolution of all example videos. The GPU memory utilization and timing results are shown in Figure 7.1. All results were run on a NVIDIA Titan X GPU.

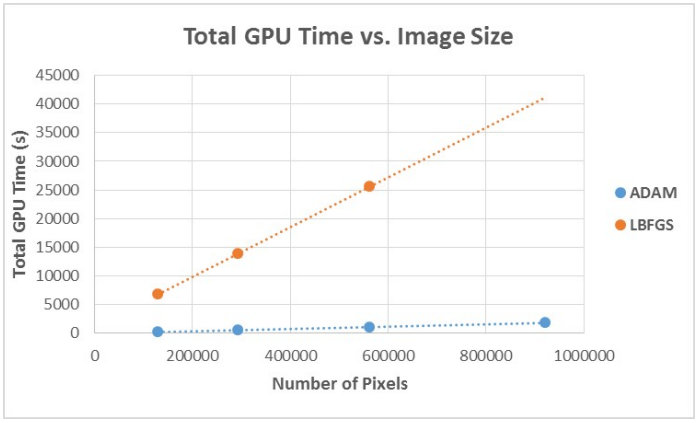
The memory results show that the baseline LBFGS method utilizes more memory than Adam and does not allow for the maximum output resolution of the original videos (1280×720). The results in Figure 7.1b show the normalized iteration timing results between the two algorithms showing that they are relatively similar in timing, however the total time results in Figure 7.1c show that there is a significant disparity between the adaptive heuristics that we use and the baseline. The adaptive heuristics reduce the total number of starting iterations to reach acceptable outputs from 1000 in the baseline to merely 100 in our method. Furthermore, iterations after the first iteration reduce this number to 50 for all subsequent iterations which significantly reduces the overall computational time. Although the baseline version can run completely in parallel (due to no temporal consistency), it would require a significant number of GPUs to achieve.



(a)



(b)



(c)

Figure 7.1: GPU utilization for NVIDIA Titan X using two different optimization schemes. LBFGS results are missing for 921600 due to overloading. Memory utilization (a). Iteration-normalized time (b). Total GPU time over 80 frames (c).

A baseline approach was used to compare our results applied the stylization algorithm one frame at a time using “The Starry Night” by Vincent Van Gogh as the reference style image. This approach had significant jitter as evident in Fig. 7.2 and 7.3 which show that localized structures such as brush strokes and colors change very rapidly between frames in this naïve approach.

The dense optical flow approach using the non-local method described by Sun et al. [86] stabilized motion significantly. Motion preservation is fairly evident in the local structures of the mountainside in Fig. 7.2b. Brush strokes are very well preserved between frames in the optical flow approach while these features move around significantly in the baseline approach. Similarly, in Fig. 7.3b, the notable feature of the woman’s ear has significantly different detail between frames in the naïve, non-motion matched approach, however the optical flow approach clearly preserves local style structures over time.

Once the amount of style transferred between frames was optimized for overall visual experience, the adaptive approaches were then introduced to increase the overall temporal coherence between frames. This came in the forms of acceleration detection, contrast adaptation, and brightness adaptation. The acceleration detection produces subtle differences, especially when looking at still frames, and as such, we have omitted figures from our results. More complete results are shown in [20].

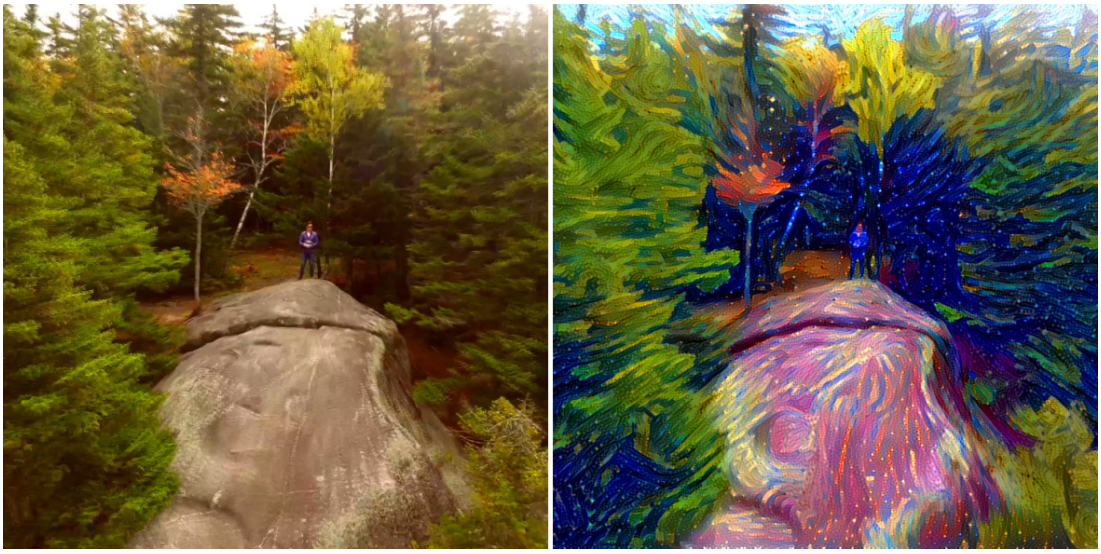
We tested our dynamic contrast adaptation by processing a video with a

rapid fade in from a completely black scene. This sequence of tests is shown in Fig. 7.4.

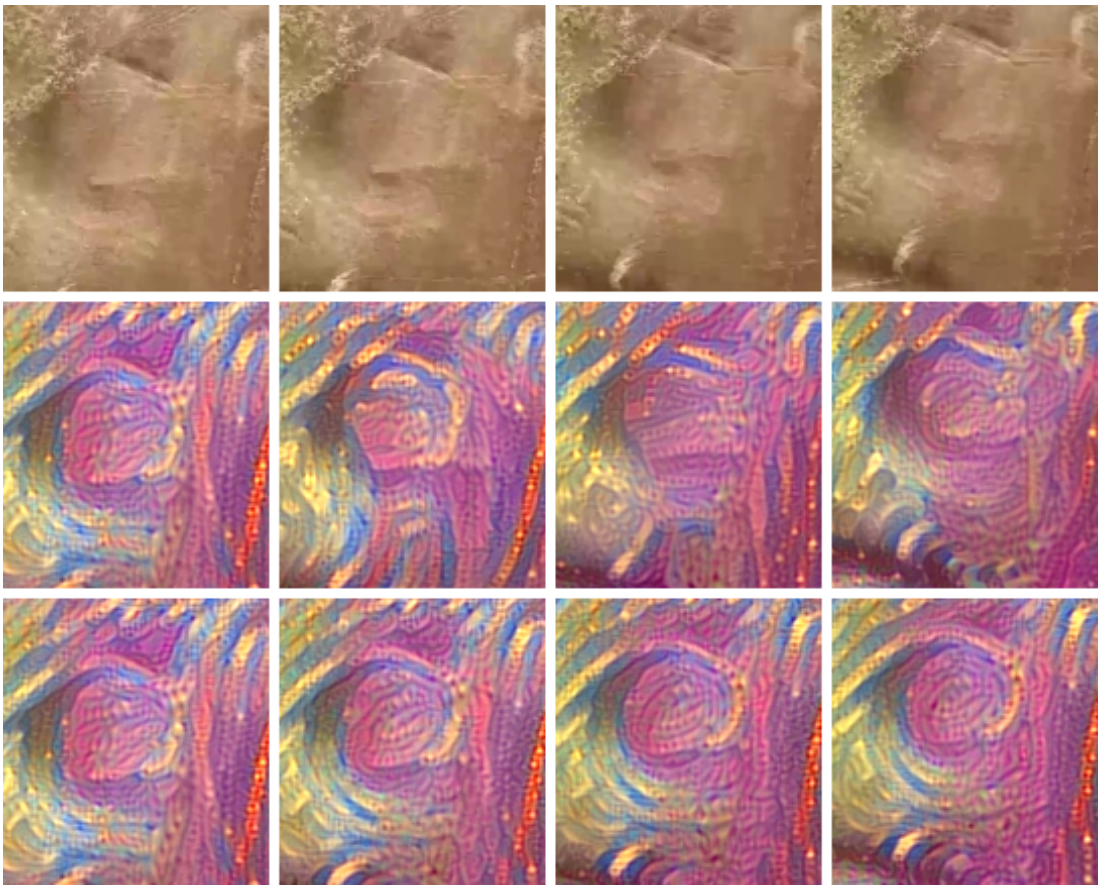
The non-adapted case has significant artifacting from incorrect style transfer from the apparent motion that is evident in the black scene transitioning into a content-rich scene. This resulted in streaking of textures in the non-adapted case which is visually unappealing. This is minimized in our adaptive case which matches the visual style and contrast of the original frames.

Next we tested our brightness adaptation which was tested by varying the amount of light in a scene until it was completely white. This fade to white can be seen in Fig. 7.5.

Both brightness and contrast are adapated in this experiment. This experiment shows that without brightness and contrast adaptations, the output frames are free to vary wildly and do not faithfully match the inputs that generated them. Further, the propagation of style through frames is not dampened and thus the non-adapted case transfers its style to a white background which can be disruptive to a smooth and continuous viewing experience.



(a)



(b)

Figure 7.2: Motion tracking comparison of “This is Vermont Foliage”. (a) Original video on the left, stylized video on the right, (b) Top to bottom: original frames, naïve approach, optical flow tracked approach.

(a)

(b)

Figure 7.3: Motion tracking comparison of “People Walking Around”. (a) Original video on the left, stylized video on the right, (b) Top to bottom: original frames, naïve approach, optical flow tracked approach.

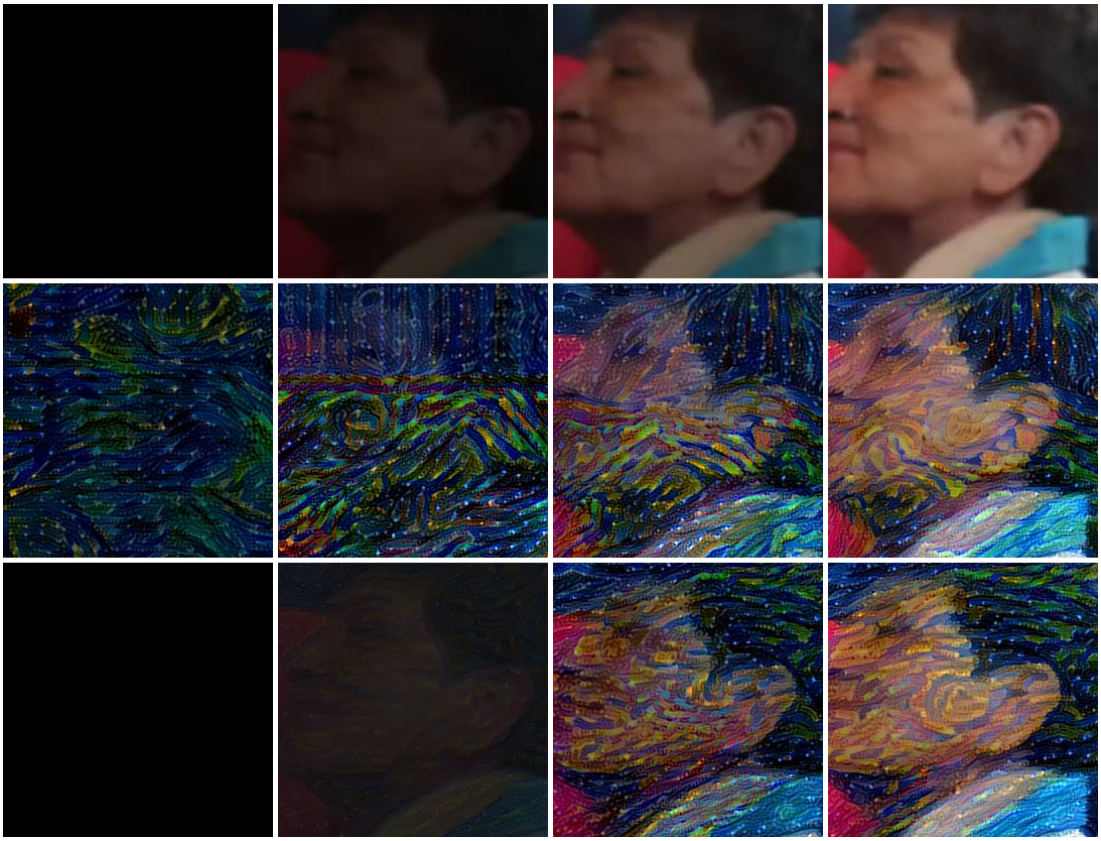


Figure 7.4: Comparison of inter-frame contrast differences. Top to bottom: original frames, no contrast adaptation, dynamic contrast adaptation.

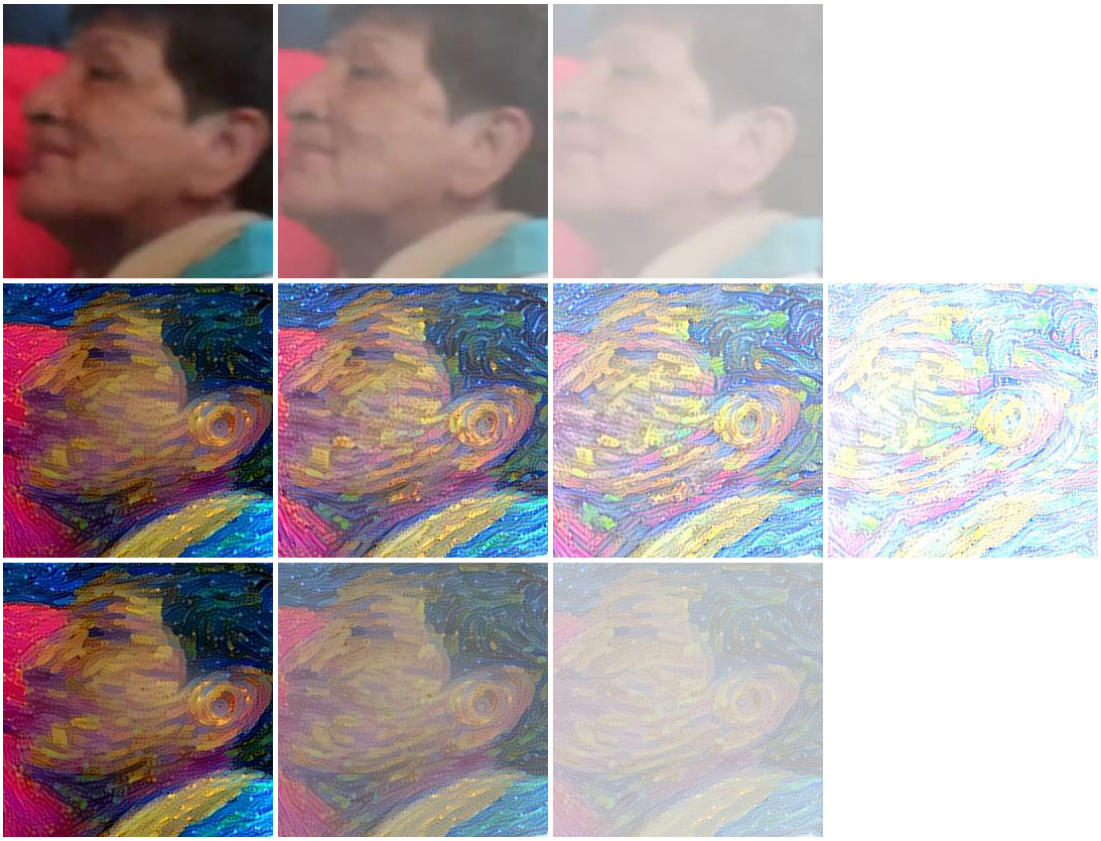


Figure 7.5: Comparison of inter-frame brightness differences for a fade-to-white. Top to bottom: original frames, no brightness adaptation, dynamic brightness adaptation.

Chapter 8

Conclusions

We have demonstrated that our spatiotemporal framework is robust enough to automatically produce temporally coherent videos that exemplify a given artistic style without distracting jittering effects. Through the use of a neural style transfer algorithm, non-local optical flow, and several adaptations, this research is able to produce videos that are well behaved and match the original temporal content under a variety of lighting conditions while preserving texture information from a reference artwork. Further, we have showed that a completely automated system for producing motion matched artistic works is entirely possible given current advances in research which can free animators of the laborious process of performing frame-by-frame stylization and allow them to focus on their own artistic style contributions instead. As such we present our framework for producing these animations as a tool that can be applied to any type of stochastically influenced stylization process that stands to benefit from temporal coherence between generated frames of a video in order to maximize viewer comfort and overall visual appeal.

Future improvements to this work include long-term temporal dependency handling and occlusion detection using [70]. Improvements to stylization can also be made using Markov random fields [60]. Another future improvement to consider is the utilization of a segmentation-based approach to stylize specific objects in a scene with different styles and to track them over time. These improvements would expand and enhance the aesthetic quality of the techniques proposed in this thesis.

Bibliography

[1] Aseem Agarwala. Snaketoonz: A semi-automatic approach to creating cel animation from video. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 139–ff. ACM, 2002.

[2] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

[3] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[4] Roberto Battiti and Francesco Masulli. Bfgs optimization for faster and automated supervised learning. In *International neural network conference*, pages 757–760. Springer, 1990.

[5] Matt Benedetto. This is vermont foliage. `https://www.youtube.com/watch?v=oH3yL84XeWo`, oct 2015.

[6] James R Bergen, Patrick Anandan, Keith J Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *Computer VisionECCV'92*, pages 237–252. Springer, 1992.

[7] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer vision and image understanding*, 63(1):75–104, 1996.

[8] Simon Breslav, Karol Szerszen, Lee Markosian, Pascal Barla, and Joëlle Thollot. Dynamic 2d patterns for shading 3d scenes. In *ACM Transactions on Graphics (TOG)*, volume 26, page 20. ACM, 2007.

[9] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision-ECCV 2004*, pages 25–36. Springer, 2004.

[10] ButForTheSky. People walking around. `https://www.youtube.com/watch?v=afb_GOx-BAU`, jul 2011.

[11] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[12] P Campolucci, F Capperelli, S Guarnieri, F Piazza, and A Uncini. Neural networks with adaptive spline activation function. In *Electrotechnical Conference, 1996. MELECON'96., 8th Mediterranean*, volume 3, pages 1442–1445. IEEE, 1996.

[13] Alex J Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *arXiv preprint arXiv:1603.01768*, 2016.

[14] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[15] John P Collomosse, David Rowntree, and Peter M Hall. Stroke surfaces: Temporally coherent artistic animations from video. *Visualization and Computer Graphics, IEEE Transactions on*, 11(5):540–549, 2005.

[16] Martin Davis. The mathematics of non-monotonic reasoning. *Artificial Intelligence*, 13(1):73–80, 1980.

[17] Michael RW Dawson and DON P SCHOPFLOCHER. Modifying the generalized delta rule to train networks of non-monotonic processors for pattern classification. *Connection Science*, 4(1):19–31, 1992.

[18] Peter Dayan and Laurence F Abbott. Theoretical neuroscience, vol. 806, 2001.

[19] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1841–1848, 2013.

[20] Michael Dushkoff. Style flow. `https://www.youtube.com/watch?v=xb0UoR9gU1U`, mar 2016.

[21] Michael Dushkoff, Ryan McLaughlin, and Raymond Ptucha. A temporally coherent neural algorithm for artistic style transfer. *Proceedings of International Conference on Pattern Recognition*, 2016.

[22] Michael Dushkoff and Raymond Ptucha. Adaptive activation functions for deep networks. *Electronic Imaging*, 2016(19):1–5, 2016.

[23] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[24] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341, 2009.

[25] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image analysis*, pages 363–370. Springer, 2003.

[26] J Fivser, M Lukavc, O Jamrivska, M Vadik, Y Gingold, Paul Asente, and Daniel Sykora. Color me noisy: Example-based rendering of hand-colored animations with temporal noise control. In *Computer Graphics Forum*, volume 33, pages 1–10. Wiley Online Library, 2014.

[27] Gary W Flake. *Nonmonotonic activation functions in multilayer perceptrons*. PhD thesis, University of Maryland, 1993.

[28] Catherine E Garabedian, Stephanie R Jones, Michael M Merzenich, Anders Dale, and Christopher I Moore. Band-pass response properties of rat si neurons. *Journal of neurophysiology*, 90(3):1379–1391, 2003.

[29] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[30] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. *arXiv preprint arXiv:1505.07376*, 2015.

[31] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[32] James J Gibson. The perception of the visual world. 1950.

[33] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275, 2011.

[34] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. Maxout networks. *ICML (3)*, 28:1319–1327, 2013.

[35] Stefano Guarnieri, Francesco Piazza, and Aurelio Uncini. Multilayer feedforward networks with adaptive spline activation function. *IEEE Transactions on Neural Networks*, 10(3):672–683, 1999.

[36] Paul Haeberli. Paint by numbers: Abstract image representations. In *ACM SIGGRAPH Computer Graphics*, volume 24, pages 207–214. ACM, 1990.

[37] Michael Haggerty. Almost automatic computer painting. *IEEE Computer Graphics and Applications*, 11(6):11–12, 1991.

[38] Kazuyuki Hara and K Nakayamma. Comparison of activation functions in multilayer neural network for pattern classification. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 5, pages 2997–3002. IEEE, 1994.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[41] John D Head and Michael C Zerner. A broydenfletchergoldfarbshanno optimization procedure for molecular geometries. *Chemical physics letters*, 122(3):264–270, 1985.

[42] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460. ACM, 1998.

[43] Aaron Hertzmann. Paint by relaxation. In *Computer Graphics International 2001. Proceedings*, pages 47–54. IEEE, 2001.

[44] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[45] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[46] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.

[47] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[48] Stephen J Huston and Holger G Krapp. Visuomotor transformation in the fly gaze stabilization system. *PLoS Biol*, 6(7):e173, 2008.

[49] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[50] Justin Johnson. neural–style. online, 2015.

[51] Miao Kang and Dominic Palmer-Brown. An adaptive function neural network (adfunn) for phrase recognition. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 1, pages 593–597. IEEE, 2005.

[52] Miao Kang and Dominic Palmer-Brown. An adaptive function neural network (adfunn) for phrase recognition. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 1, pages 593–597. IEEE, 2005.

[53] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[54] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[55] Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. State of the ”art”: A taxonomy of artistic stylization techniques for images and video. *Visualization and Computer Graphics, IEEE Transactions on*, 19(5):866–885, 2013.

[56] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[57] Hochang Lee, Sanghyun Seo, Seungtaek Ryoo, and Kyunghyun Yoon. Directional texture transfer. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 43–48. ACM, 2010.

[58] Ruen-Rone Lee. A colored pencil non-photorealistic rendering for 2d images.

[59] Victor Lempitsky, Stefan Roth, and Carsten Rother. Fusionflow: Discrete-continuous optimization for optical flow estimation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[60] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. *arXiv preprint arXiv:1601.04589*, 2016.

[61] Xiaofeng Liao, Kwok-Wo Wong, Chi-Sing Leung, and Zhongfu Wu. Hopf bifurcation and chaos in a single delayed neuron equation with non-monotonic activation function. *Chaos, Solitons & Fractals*, 12(8):1535–1547, 2001.

[62] Peter Litwinowicz. Processing images and video for an impressionist effect. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 407–414. ACM Press/Addison-Wesley Publishing Co., 1997.

[63] Barbara J Meier. Painterly rendering for animation. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 477–484. ACM, 1996.

[64] Cory Merkel, Dhireesha Kudithipudi, and Nick Sereni. Periodic activation functions in memristor-based analog neural networks. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–7. IEEE, 2013.

[65] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2012.

[66] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[67] Kenji Nakayama and Moritomo Ohsugi. A simultaneous learning method for both activation functions and connection weights of multilayer neural networks. In *Proc. IJCNN*, volume 98, pages 2253–2257, 1998.

[68] László Neumann and Attila Neumann. Color style transfer techniques using hue, lightness and saturation histogram matching. In *Computational Aesthetics*, pages 111–122. Citeseer, 2005.

[69] Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

[70] Eduardo Parrilla, Damián Ginestar, José Luis Hueso, Jaime Riera, and Juan Ramón Torregrosa. Handling occlusion in optical flow algorithms for object tracking. *Computers & Mathematics with Applications*, 56(3):733–742, 2008.

[71] F Piazza, A Uncini, and M Zenobi. Artificial neural networks with adaptive polynomial activation function. 1992.

[72] Vassilis P Plagianakos, George D Magoulas, and Michael N Vrahatis. Deterministic nonmonotone strategies for effective training of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 13(6):1268–1284, 2002.

[73] VP Plagianakos, DG Sotiropoulos, and MN Vrahatis. A nonmonotone backpropagation training method for neural networks. *Dept. of Mathematics, Univ. of Patras, Technical Report*, (98-04), 1998.

[74] Michael JD Powell. Radial basis functions for multivariable interpolation: a review. In *Algorithms for approximation*, pages 143–167. Clarendon Press, 1987.

[75] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer graphics and applications*, (5):34–41, 2001.

[76] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Computer Vision and Pattern Recognition*, 2015.

[77] S Roth, JP Lewis, D Sun, and MJ Black. Learning optical flow. In *ECCV*, pages 83–97, 2008.

[78] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos, 2016.

[79] Gabriele Scheler. Memorization in a neural network with adjustable transfer function and conditional gating. *arXiv preprint q-bio/0403011*, 2004.

[80] Gabriele Scheler. Regulation of neuromodulator receptor efficacyimplications for whole-neuron and synaptic plasticity. *Progress in Neurobiology*, 72(6):399–415, 2004.

[81] Matt Silverman. History of rotoscoping, jun 2004.

[82] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[83] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[84] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[85] Michael Soltiz, Dhireesha Kudithipudi, Cory Merkel, Garrett S Rose, and Robinson E Pino. Memristor-based neural logic blocks for nonlinearly separable functions. *IEEE Transactions on computers*, 62(8):1597–1606, 2013.

[86] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE, 2010.

[87] Lorenzo Vecci, Francesco Piazza, and Aurelio Uncini. Learning and approximation capabilities of adaptive spline activation function neural networks. *Neural Networks*, 11(2):259–270, 1998.

[88] Andreas Wedel, Thomas Pock, Christopher Zach, Horst Bischof, and Daniel Cremers. An improved algorithm for tv-l 1 optical flow. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 23–45. Springer, 2009.

[89] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013.

[90] K-W Wong, C-S Leung, and S-J Chang. Use of periodic and monotonic activation functions in multilayer feedforward neural networks trained by extended kalman filter algorithm. *IEE Proceedings-Vision, Image and Signal Processing*, 149(4):217–224, 2002.

[91] Jonas Wulff and Michael J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) 2015*, June 2015.

[92] Xuezhong Xiao and Lizhuang Ma. Gradient-preserving color transfer. In *Computer Graphics Forum*, volume 28, pages 1879–1886. Wiley Online Library, 2009.

[93] Shuxiang Xu and Ming Zhang. Justification of a neuron-adaptive activation function. In *IEEE-INNS-ENNS international joint conference on neural networks*, pages Vol3–465, 2000.

[94] Hanzhong Ye. Crowdsourced rotoscoping. Master's thesis, EECS Department, University of California, Berkeley, Jun 2012.

[95] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[96] Chien-Cheng Yu, Yun-Ching Tang, and Bin-Da Liu. An adaptive activation function for multilayer feedforward neural networks. In *TENCON'02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, volume 1, pages 645–650. IEEE, 2002.

[97] W. Zhang, S. Xiao, and X. Shi. Low-poly style image and video processing. In *Systems, Signals and Image Processing (IWSSIP), 2015 International Conference on*, pages 97–100, Sept 2015.

[98] Yinda Zhang, Jianxiong Xiao, James Hays, and Ping Tan. Framebreak: Dramatic image extrapolation by guided shift-maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1171–1178, 2013.