QUASI-SEMANTIC IMAGE MANIPULATION VIA DEEP NEURAL NETWORKS

by

JOHN LEELAND GIBBS

(Under the Direction of Frederick W. Maier)

ABSTRACT

This thesis explores two uses of deep neural networks to perform quasi-semantic visual tasks in two domains: creating convincing color correction for raw video stills, and discovering images semantically similar to a search image. First, two methods are compared to determine which works better for color grading. Method A, using a retrained classification network, works very effectively but is not easily extensible. Method B, using a trained conditional Generative Adversarial Network, works extremely well, though it softens images a small amount and can create artifacts in the corrected images. Of the two methods, cGAN is chosen as the best option for future research. Second, we repurpose and retrain a classification network to create a histogram of output classifications used to recall images that are similar to a query image. This method works effectively, discovering images that match or nearly match the query image with a high degree of precision.

INDEX WORDS: Artificial Intelligence; Neural Network; Deep Neural Network; Convolutional Neural Network; CNN; Image Based Recall; Conditional Generative Adversarial Network; cGAN; Color Correction; Color Grading

QUASI-SEMANTIC IMAGE MANIPULATION VIA DEEP NEURAL NETWORKS

by

JOHN LEELAND GIBBS

B.A., Princeton University, 1987

M.A., The Ohio State University, 1990

Ph.D., The Ohio State University, 1995

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2019

QUASI-SEMANTIC IMAGE MANIPULATION VIA DEEP NEURAL NETWORKS

by

JOHN LEELAND GIBBS

Major Professor:	Frederick Maier
Committee:	W. Don Potter
	Khaled Rasheed

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
May 2019

DEDICATION

For my wife, Lane, and my boys, Joshua, Kenlee, Hunter, and Hayden.

ACKNOWLEDGEMENTS

I would like to thank my committee for their support, encouragement, and hard work teaching an old dog new tricks. Dr. Frederick Maier, who directed this thesis and has been more than a mentor to me over the past several years, has been instrumental in any success I have had in this field. I have benefitted greatly from your classes, your sage advice, and your friendship. Thank you also to Dr. Don Potter, who introduced me to the Artificial Intelligence program, and taught several classes which really stretched my abilities, and my understanding of what AI truly was. Thank you, too, Dr. Khaled Rasheed, for the hard work you put in to working with me in class and out. Without your direct support, I could never have gotten to this point.

A big thank you as well to my home department: the Department of Theatre and Film Studies. The faculty, staff and students there, and especially Dr. David Saltz, department head, have been wonderfully flexible and understanding as I have pursued this degree. A thank you is also due to the University of System of Georgia's Tuition Assistance Program, which made this degree financially possible.

I also wish to thank my parents, Lee and Joan, for instilling within me a love of life-long learning. This work—and frankly my entire career path—is due more to your influence in my young life than anything else.

Finally, I would like to thank my wife, Lane, for her tireless and enthusiastic support of my studies. Your love, and your confidence in my abilities kept me going during some difficult times, and allowed me to finish this program and this work. Thank you!

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1 Introduction

Recently a great deal of excitement has focused on so-called deep neural networks and the ability to train them to perform highly complex tasks—tasks that verge on the semantic in that an artificial neural network can process images, words, and other items in a quasi-meaningful manner. While they are (of course) not generally knowledgeable, these neural networks do an impressive job at the specific task for which they have been trained, and research is being conducted into use of these networks for more and more diverse uses.

This thesis consists of two papers, published recently, that explore different methodologies that utilize deep neural networks (specifically convolutional networks) to 'recognize' images and manipulate data associated with them. Chapter 2 combines two related publications into one chapter. The conference paper, "Aesthetic Grading: Color Correction via Neural Networks," was presented at the Theory and Practice in Modern Computing conference (part of the Multi-Conference on Computer Science and Information Systems) in Madrid, Spain, in July, 2018, and published in the conference proceedings (Kundert-Gibbs 2018). This paper was selected as one of only eight papers in the conference to be extended and published separately. The extended version, "Video Color Grading via Deep Neural Networks," was published in the International Journal on Computer Science and Information Systems, 13, No., 2 (Gibbs 2018). Chapter 3 is an extended version of the paper, "Image-Based Content Retrieval via Class-Based Histogram Comparisons," presented at the IT convergence and security conference in Seoul,

South Korea, in September of 2017, and published in the conference proceedings (Kundert-Gibbs 2017).

Chapter 2 compares the efficacy of two neural network types—classification and conditional Generative Adversarial—in their respective abilities to resolve image problems commonly found in raw video sequences. Chapter 3 demonstrates the use of a novel histogram comparison method that is utilized to recognize best matches to a query image and to return them to the user. Together, these chapters exemplify the breadth of cases for which deep convolutional neural networks can be utilized. Chapter 4 briefly presents the import of the work done in the thesis as a whole, as well as some discussion of research that has occurred since the publication of the two articles.

1.2 Summary of Research

We here briefly summarize the results presented in chapters 2 and 3. Chapter 2 compares two different methods to solve image correction issues—commonly termed color grading or color correction—in video sequences. The first method explored is to retrain a classification network to recognize specific color correction issues exemplified in a given image (or image sequence). The goal of using this system would be to identify a problem or problems, and guide another piece of software to make corrections to the image(s). Generally speaking, color grading issues are of a low frequency nature, such as the wrong color cast (e.g., too green), too much contrast, or an incorrect black point (e.g., set 33% too high). In other words, rather than high frequency elements—like edges, corners, and rapidly changing color—that most classification networks are trained to recognize, our retraining is specifically related to the low frequency aspects of an image: its general color, overall contrast between light and dark areas, and the general brightness of light areas versus dark ones. Retraining a Convolutional Neural Network

(CNN) to detect these image issues is extremely effective, even though the original network was not designed to train for such low frequency image elements. The two big, related problems with our classification method are data labeling and extensibility. It turns out that the network over-trains very easily (and this is not readily resolved) and thus new problems, or combinations of problems, are not properly classified. This shortcoming necessitates a very large, varied, and labeled data set, which would be extremely time consuming to create. Additionally, there is a more fundamental problem in classification itself: how does one classify an image with potentially dozens of changes made on a more-or-less infinite scale? For example, an image might be 22% too blue, have a black point 15% too low, a contrast 64% too high, and an orange cast that is 71% too much. With so many variables, creating a complete (or even nearly complete) classification database could require tens of thousands of categories, which would reduce the efficacy of the network and also make labeling the thousands or even millions of images needed for training extremely challenging.

The second method explored for image correction is a conditional Generative Adversarial Network (cGAN). This relatively new type of neural network actually changes the pixels in an input image to create an output image (rather than simply classifying image categories or problems like a classification network). One well known use for this type of network is to colorize black and white photographs (see Iizuka, Simo-Serra, and Ishikawa 2016): a black and white input image is presented to a cGAN, which outputs a color image that should be indistinguishable from an actual color photo. Using this system for color correction in a video sequence produces excellent results, creating images that are nearly equivalent (on a pixel-by-pixel basis) to properly color graded images (see Figure 2.7 for an example). Furthermore, this network can be trained with ‘before and after’ images—in a semi-supervised mode—that do not

need to be labeled; thus any sequence of images can be utilized without the need for anyone to label the data, nor even need to explicitly know what is wrong with the 'before' images. The drawbacks of this network are that it can create artifacts in the resultant image (as an entirely new image is being produced), that it softens the resulting image slightly, and that image sequences can appear to flicker, as each image is produced independently. All three of these drawbacks are mitigated in the results section of Chapter 2. Our conclusion is that, while both networks perform very well, the cGAN is the method that should be explored further. Its drawbacks are smaller and are easier to mitigate, it produces excellent output images, and it can be further trained on very large datasets without the need to label any data.

In Chapter 3, we find that our histogram-based use of a retrained classification network is effective at finding images in a database that are semantically similar to a query image. Our novel approach repurposes the standard image-net classification system, which consists of 1,000 categories. Rather than the nominal use of a classification network to discover a best category for an image by extracting the category (from the 1,000 candidates) with the highest probability, we examine the query image's histogram, created by all 1,000 probabilities, and match this with the most similar histograms from our image database. With some reworking, some retraining, and a voting system (all described in Chapter 3), the histogram method works remarkably well at finding images that are qualitatively very similar to given query images. Though there is a one time startup penalty in that every image in the search database has to have its histogram calculated, the actual comparison of histograms is very straightforward and can be accomplished rapidly. Also of interest, when a query image is input that has 0 matches in the search database, the network performs a 'graceful failure,' finding images that are extremely similar even though they are not the same (as they cannot be, given that there are no matching images in the

database). As the classification network was never designed to be used for image-based recall, nor was it designed for the histogram of results to be significant, the success of this method is remarkable.

While chapters 2 and 3 are self-contained, as they were published independently, we will take the opportunity in the rest of the introduction to discuss more general matters that help contextualize the work presented later in this paper. We present a brief, basic overview of the history of neural networks, then discuss Convolutional Neural Networks (CNNs), and finally discuss the newer Generative Adversarial Networks (GANs) and conditional Generative Adversarial Networks (cGANs).

1.3 Introduction to Neural Networks and Convolutional Neural Networks

Neural networks have been around in one form or another since the early days of Artificial Intelligence research. Even as far back as Alan Turing's work in the 1930s and 40s (see Turing 2004 for several of his seminal articles), he noted the possibility that a computing device could hypothetically operate similarly enough to a human to fool the test subject—an experiment now commonly referred to as the Turing test. Though this work was purely theoretical, and did not involve neural networks, it nonetheless established the notion that computers could somehow model the human mind. On a more practical level, Grey Walter in the late 1940s invented Elmer and Elsie, two autonomous robots with rudimentary vision circuitry.

1.3.1 Early Neural Networks

It was Frank Rosenblatt's work at Cornell University in the late 1950s that created the first true artificial neural networks, which Rosenblatt named perceptrons as his initial work was on vision. These rudimentary neural networks, which consisted of a complex blend of digital and analogue parts, were able to distinguish basic shapes, such as triangles and squares. More importantly,

perceptrons demonstrated that Rosenblatt's theory—based on his work in Psychology—that one could model the neuronal thresholding in electronic circuitry was viable. Though these early perceptrons, shown in Figure 1.1, only worked on simple problems, they could in fact 'learn' via a shifting of input weights of a node. Each node would take its inputs and, using a non-linear function—initially utilizing physical stepping motors(!)—would 'fire' if enough input signal had been detected: a threshold had been reached. This thresholded cellular firing (via action potential in cellular sodium channels) is how neurons in an animal brain work, and Rosenblatt's ability to vary input weights of a perceptron node was analogous to reducing the transmission resistance between an axon and a dendrite in a biological brain.

Figure 1.1: Frank Rosenblatt (right) working on an early perceptron (from the Pace University archives)

A basic three layer perceptron is shown in Figure 1.2. As can be seen, there is an input layer, which takes in some signal from the outside world (e.g., data from a vision system), an output layer, which generally makes a decision concerning the input (e.g., is it a triangle or a square?), and a hidden layer of nodes that take the input data and either fire (pass on the signal) or do not fire (ignore the incoming signal) depending on how much activity they receive from the nodes in the previous layer. Weighted connections join all of these neuron-like nodes, functioning like the axon-dendrite pair in a biological brain. Generally there is a bias weight (a constant-on signal that is added no matter what) and a weight that is multiplied by the incoming signal from connected nodes. Both bias and multiplicative weights can be positive (excitatory) or negative (inhibitory), and both can vary between 0 and an arbitrary number, though normally values are kept within a reasonable range of <= ~3 digits (e.g., less than 100 or so). Both bias and multiplicative weights are changed via feedback as they are trained. In current practice, when trained with a labeled data set, networks use a feed-forward, back-propagating loop to alter weights (e.g., see Walker 1993): a signal is passed from input, via all connected nodes, to output, producing a decision (e.g., input is a triangle). If the decision is correct (e.g., the data is labeled a triangle) weights are increased at each node connection by using chained partial differential equations to determine how much the weights should be changed. If the answer is wrong (e.g., the data is labeled a square), then weights are reduced in the same manner to account for this error. A learning rate < 1.0 is used to reduce the effect of each data point, so that the network makes only small changes to weights, not large ones, which could overshoot proper weighting of the network. Running the network through all the training data (e.g., 10,000 training images) is called an epoch, and generally after each epoch, the network's performance is tested on a separate data set called a testing data set. This test data provides a quantified evaluation of the

network's performance at that point. Finally, to fully validate the quality of the network, a completely withheld validation data set is used after training to test the network against completely unseen examples. From a master data set, one might choose to use 70% of the data for training, 20% for testing, and 10% for validation.

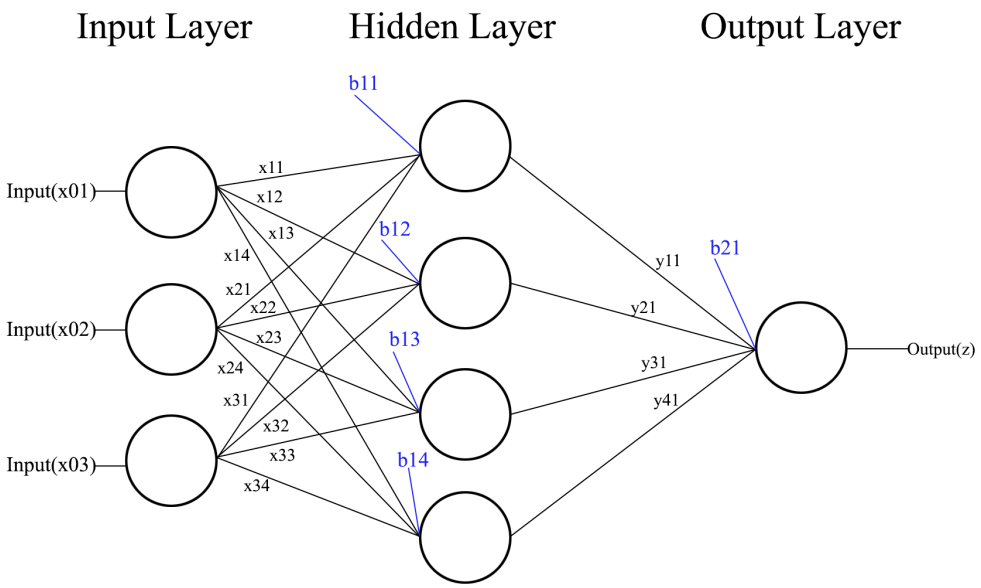

Figure 1.2: Schematic of a basic three layer perceptron

Much work was done on perceptrons through the 1960s (see for example, Rosenblatt 1957, 1960, 1966), but then progress slowed and AI work turned to other arenas. In the 1980s, with the widespread availability of fairly powerful digital computers, there was a resurgence in work on artificial neural networks (see for example, LeCun 1988 and 1989, Hinton 1989). Of particular note, in the later 1980s were the first real experiments with 'deep' neural networks—networks that had more than one or two hidden layers (see for example, Tano 1982, Pessa 1988, Werbos 1992). While significantly more complex, these networks showed the promise of being able to learn fairly complex, real-world tasks. Researchers like Geoffrey Hinton and Jan LeCun (both

recipient of the 2018 Turing Award), who are now famous in AI circles, did their initial work during this period.

While promising initially, research into these early deep neural networks stalled out due to three main reasons: lack of processing power, lack of huge data sets, and the overall complexity of a deep neural network. Computers at the time were simply incapable of dealing with the huge numbers of calculations necessary to run a substantial neural network in decent time, and computer memory was also limited, reducing the number of neurons that could be kept in memory simultaneously. In addition, producing and labeling a large enough digital data set to train these networks was beyond the scope of most research at the time. Projects like the NIST (later to become the MNIST) data set of (originally) 4,000 images of hand written digits, in 1995 (Grother 1995) was a major step forward in creating large enough digital data sets that could be used to train neural networks. As for complexity, fully connected deep neural networks have so many connections that the overall complexity of the network explodes exponentially, multiplying the computing power requirements in a commensurate manner. Progress in deep neural networks continued into the early 2000s, but interest in general waned, and the pace of progress was slow.

Major changes came late in the first decade of the 21st Century. Computing power and memory improved drastically during this time, but just as importantly, huge new digitized datasets were being created, and changes in network architecture, especially the creation of Convolutional Neural Networks, or CNNs came onto the scene.

1.3.2 Deep Convolutional Networks

New image databases consisting of over one million images, like the image-net data set (image-net 2018) provided deeper neural networks, with thousands of nodes, with the number and variety of input images they needed to be properly trained. Second, using convolutional layers

near the beginning of a deep network was a major breakthrough. These new networks, termed Convolutional Neural Networks, or CNNs, utilize three or more convolutional layers—where numerous (initially random) convolutional filters are passed (convolved) over the image—combined with pooling and renormalizing layers, to extract higher level features from the input image. These features are finally passed to one or more traditional, fully connected layers, which produce the final results. CNNs reduced the complexity of deep neural networks because the convolutional (and pooling and renormalizing) layers only connect in a localized manner, rather than being fully connected with every other node in the next and previous layers of nodes. This reduced the number of weights that needed to be tracked, and thus the complexity of these networks to a point where contemporary computers—especially ones that utilize the new power of computation via graphics cards—can train networks in a reasonable time.

For each node in a CNN, a non-linear thresholding function is used to determine whethere the node fires or not. One further refinement in modern CNNs is that the traditional Sigmoid function ($S(x) = \frac{}{1+}$) is replaced with a Rectified Linear Unit, or ReLU, function ($S(x) = \max($). Figure 1.3 shows graphs of the two functions for comparison. While the ReLU is technically not admissible as a non-linear function, as it is not differentiable (there is a discontinuity at 0), it is much more effective in training the network, and it is enormously faster (as it simply determines if a number is greater than 0) and thus allows these huge networks to run at a reasonable speed. Thus in practice ReLU is commonly used, even though it technically violates the back propagation methodology as it is not properly differentiable. In standard practice, the derivative of 0 is specified to be 0, which resolves the differentiation issue.

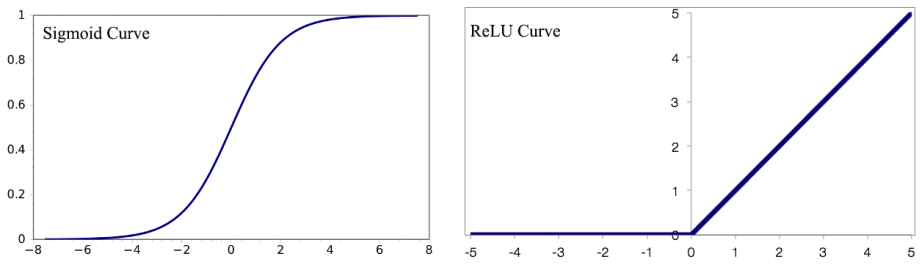

Figure 1.3: A sigmoid curve (left) versus a Rectified Linear Unit (ReLU) curve (right)

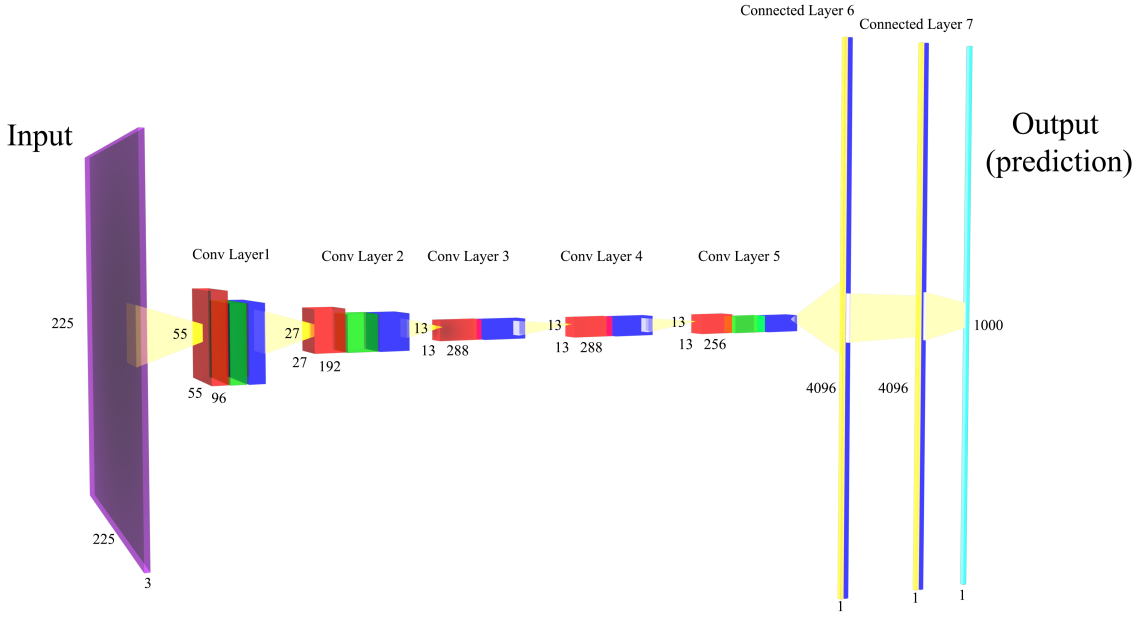

Figure 1.4: A schematic of the architecture of a deep convolutional neural network, or CNN

As Figure 1.4 shows, a classification type CNN generally reduces the dimensionality of the problem to a single dimension at the end. The input matrix is most often a square image of, say, 224 by 224 pixels. Each pixel of the image (50,176 in the case) is an input node. Information passes from this through the multiple hidden layers to the final layer, which is a one-dimensional vector (1 by 1,000 in this case). This final vector, which consists of predictions (e.g., dog, cat, bird, fish, etc.) consists of a set of floating point numbers, and the largest value is the prediction the network makes (e.g., there is a 0.87 probability that this is a picture of a dog). Between input

and output are numerous nodes that reduce the width and height dimensionals of the data, while increasing the depth dimension. Most often in the convolutional portion of the network, there are collections of layers in the following group: convolution layer, pooling layer, normalization layer. The convolution layer convolves a filter over the input image (or input data at later stages). If the filter is 3X3, nine weights, organized into a square pattern like a tic-tac-toe board, are passed over the values of the input data, multiplying the values and summing them up into a new, single value, and thus reducing the dimensions of the original image. Next, a pooling layer will often pool values (using an algorithm like max pooling), thus further reducing the input data's dimensions. Finally, a normalization layer will renormalize the data values so that outlying values are suppressed (outlying values can cause an entire network to depend on just a few nodes, rather than the totality of nodes).

Many convolutional filters are applied to the input data, so while the width and height dimensions of the input shrinks, the depth dimension rises from 3 (for a color input image) to 32 or 64 or even higher as the layers get deeper. Normally the three-layer sequence (convolution, pooling, renormalizing) is repeated several times in the convolutional stage of the network. Finally, approximately three fully connected 'training' layers are added to the end of the network. These layers are laid out like the simpler perceptron, with a final output vector of numbers that are used to make a prediction. If the prediction is wrong, weights are reduced throughout the network—again using chained partial derivatives—and if the prediction is correct, weights are increased—just as in the simpler perceptron—using a feed-forward, back-propagation method.

The effect of the convolutional layers in CNN is for the network to extract more and more high-level features, as shown in Figure 1.5. In the first convolutional layer shown, simple patterns

have been discovered via the multiple convolution filters being trained. In the second image, higher level image elements are apparent. In the third, very high level elements have been extracted from the data. The training layers then use these last, very high level, abstracted features, to make a prediction concerning the input image.

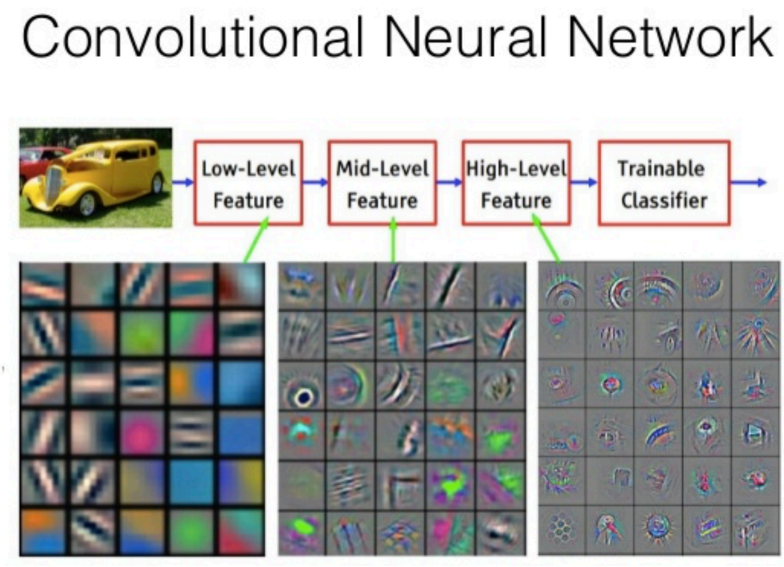

Figure 1.5: Feature extraction that occurs in layers of a CNN

1.3.3 Generative and Conditional Generative Adversarial Networks

Generative Adversarial Networks, or GANs, and conditional Generative Adversarial Networks, or cGANs, modify the standard CNN, described above, in an interesting manner: rather than use an image as input to create an output prediction (i.e., a one-dimensional vector of floating point numbers), GANs and cGANs take input pixels and create an image (i.e., a two-dimensional matrix where each value is a pixel value[1]) rather than a simple numerical prediction. For both the GAN and cGAN, two networks are trained simultaneously in competition with each other: a generative network that is responsible for creating an output image, and an adversarial network

[1] In fact both input and output images are generally 3 color RGB images, so the input/output images will be X by Y by 3, or three dimensional matrices. To simplify the explanation here, however, we present the image as a flat plane rather that three stacked planes.

that judges the quality of the produced image versus a true image from a database. The generative network in GANs takes as input some form of 'noise': in other words, a vector (generally) of randomly generated pixel colors. The generative network in cGANs, on the other hand, takes an image as input with the intent that it will produce an output image that is matched to the input image, but altered in some defined manner (directed by the training target images). Both the GAN and cGAN are trained via a database of images, but the cGAN uses image pairs: normally the input image is deficient in some manner (e.g., black and white) while the target image is correct (e.g., in color). The goal of a GAN is to produce a believable, but fictional image—like a human face that looks real, but is of no real person. The goal of a cGAN is to alter an input image to such an extent that the output is indistinguishable from a ground-truth target image—like changing a black and white image into a believable color image. For Chapter 3, we make use of the cGAN network type to rework uncorrected images so that they match corrected target images. Research into GANs and cGANs is currently at the forefront of deep neural network research.

In the following two chapters, we will discuss a way to use a cGAN to correct raw input data from a video sequence, as well as a new method of utilizing a CNN to retrieve images similar to a query image. This brief introduction to these related network types should serve as a basis from which to better understand the work discussed below.

CHAPTER 2

VIDEO COLOR GRADING VIA DEEP NEURAL NETWORKS[2]

2.1 Introduction

Color grading, also known as color correction, is a task which many film and video viewers do not even know takes place. This job is nonetheless supremely important to the professional look of a finished film or video. Color correction is the job of taking raw footage from a video/film shoot and adjusting elements such as exposure, saturation, contrast, black point, white point, and color casts to achieve a higher quality, more pleasing, and more uniform look for takes shot under different lighting conditions and on different days. While the general public might not understand that continuity and look problems exist under the controlled conditions of a professional shoot, a scene is often shot over many hours, or even several days. Thus, elements such a sunlight and/or artificial lighting can change (or the crew can move lights to better light an individual close-up shot, for example). Additionally, traditional film as well as digital camera sensors can respond differently to the same lighting conditions depending on many factors, including film chemistry changes and how long a digital camera has been running (and thus how hot the sensor is). Thus even with a trained and knowledgeable crew, there will be differences between shots in a given scene, and most assuredly there will be differences between different scenes.[3] The art of color grading and correction is to make every shot look 'good' (an

[2] This chapter is based on the following two published papers: (Kundert-Gibbs 2018) and (Gibbs 2018).

[3] In addition to the issues noted above, much professional and pro-sumer video is currently shot using a log format (which encodes raw data on a log rather than linear curve, thus allowing for more data per channel, at the expense of looking very washed out when viewed directly). Color correction, then, also involves running raw video data through established Color Look-Up Tables, or CLUTs, as a first step. As this step is already well understood and automated, it is outside the scope of this paper.

admittedly aesthetic judgment) and also to hide the differences between the various shots of a piece, which can number in the thousands for a full-length movie. The job of color grading requires a good deal of operator expertise, time, and expensive equipment, and thus costs a large amount of money—upwards of $10,000US for an independent film (Liftgammagain 2015), and substantially more for a large commercial production. Thus color correction can prove to be a significant cost for a small film, or even a large budget one. Even more critically, the expertise involved in performing the task of color grading is beyond the knowledge, budget, or time of most amateur filmmakers, YouTube authors, home video makers, and so on. For such people, color correction is not well understood and the job is often not undertaken at all, creating video output that looks amateur: elements like contrast, color casts, black point, and so on, are not adjusted (or not adjusted properly), and there is little continuity between shots.

Given these problems, professional and amateur filmmakers would find an automated solution to color correction a welcome addition, both for cost savings and for the ability to have a one-click solution to a complex and time-consuming task. While the 'artistic' task of correcting color to please the human eye, as well as to hide discontinuities in coloring for different shots, is at the same time subtle, aesthetic, and fuzzy (i.e., not obviously deterministic), and thus seems an unlikely domain for computers, we show here that color grading/correction is a process which consists of many precise steps that can be learned and executed well by either of two different neural network architectures. During a color correcting session, a color grader makes a number of traceable, quantitative steps to achieve the artistic goals of creating an intended look and hiding the variations between shots in a film. As we show here, these steps from an input (uncorrected) image to an output (corrected) image can in fact be learned by neural networks, both as a classification problem, via a classification network, and as a generative problem, via a

conditional generative adversarial network. In each case, the network produces excellent quality output, though each has ongoing issues that are the subject of continuing research.

2.2 Background

Though deep neural networks have been studied since the 1980s (Hinton 1989, Olshausen and Field 1997), improvements in computer speed, GPU speed and memory, and better algorithms that take advantage of the processing power of newer CPUs and GPUs generated a renaissance in deep neural network research by the early 2000s (Hinton and Ruslan 2006, Yoshua and Lecun 2007). When a convolutional deep neural network (CNN) won the image-net 2012 competition by a large margin (Krizhevsky, Sutskever and Hinton 2012), researchers at large noticed, and since that point a veritable flood of new research has been published utilizing deep neural nets and CNNs to great success. From understanding words (Mikolov et al 2013), to image recognition (Ciresan, Meier and Schmidhuber 2012, Zisserman 2014, Glorot, Bordes and Bengio 2011), to image caption generation (Vinyals et al 2015), to image-based recognition (Kundert-Gibbs 2017), to colorizing black and white images (Iizuka, Simo-Serra and Ishikawa 2016, Reinhard et al 2001, Zhang, Isola and Efros 2016), even to generating bizarre new images (Evans 2016, Computerphile 2016), deep neural networks have, in only a few years, come to be the preferred architecture for numerous tasks that people once considered beyond computer Artificial Intelligence ability. It is the combination of precision (of feature recognition and discrimination, for example) with the 'human' quality of understanding large-scale semantic elements in images (Chen et al 2015, Gatys, Ecker and Bethge 2016) that is particularly important to the project of color correction. In the previous chapter, we explore the semantic issue of image-based recall (IBR) (also see Kundert-Gibbs 2017) by building off a classification network (Vedaldi, Lenc and Henriques 2016, Veldaldi and Zisserman 2017). For

color correction as well, a classification network is an obvious contender, as the task bears some underlying similarities to IBR. By learning to classify ‘what is wrong’ with an image, a classification network can, via a plug-in, instruct a dedicated color grading program like DaVinci Resolve to do the actual color correction at its guidance. A contrasting method of color grading we examine is the relatively newly developed conditional generative adversarial network, or cGAN (Mirza and Osindero 2014). This network constructs entirely new, hopefully improved images, using unlabeled raw-corrected image pairs to train the network. We modified the network described in (Isola et al 2017) to work on the color correction problem, focusing training on low frequency, often subtle details in the images. The two network architectures examined here have complementary advantages and disadvantages, which are discussed below.

As with most neural network problems (and indeed modern AI as a whole), asset collection is a significant issue. Neural networks prefer large data sets, and classification networks require labeling—their major disadvantage compared to cGANs, which can learn in a semi-supervised setting (Radford, Metz and Chintala 2015). Film, fortunately, produces an almost limitless number of frames (still images which make up a movie), and thus data can be produced. There are, however, two issues: the first is that one needs matched sets of uncorrected and corrected images to train on; the second is that, at least for classification networks, these images must be labeled in a manner that captures the problems inherent in each image. For our initial work, we needed a small-to-medium-sized data set of at least 10,000 uncorrected images, each of which needed to have a corresponding corrected image as well as proper labeling to indicate the issue with the uncorrected image.

Josh Kundert-Gibbs, professional cinematographer and color grader, was able to provide us with properly adjusted and logged sample images in the following manner. He properly color graded a

number of shots—mostly talking heads from a documentary he was shooting—providing 675 frames broken down as follows: 15 different "shots" (one person talking) of 45 frames each. This set of images is 'correct' in the sense that a domain expert deems them to be so—an artistic judgment, obviously. While the judgment is artistic/aesthetic, many of the elements of good color correction, such as a good black point value, and 'not too green,' can be determined fairly effectively, if qualitatively, by looking at the shots. Beyond this, the value judgment that the shots look 'good' according to a professional's opinion is something that knowledge engineers are familiar with: inputs and outcomes are often somewhat fuzzy when learning from big data (McClean, Scotney and Shapcott 2000, Wood and Antonsson 1989). For these reasons, though 'properly color graded' is the opinion of one individual, it is a professional opinion and thus can be respected for our training/testing data set. One could eventually train duplicate networks to color grade based on different individuals' tastes, producing a number of different looks for a movie that a user could choose between.

To provide the 'uncorrected' (or detuned) images, at our direction the domain expert next created a number of carefully controlled incorrect images as follows. He took the 675 'perfect' images and detuned them via Davinci Resolve (color grading software), creating 24 sets of images (675 in each set, to match the perfect set), each of which sets has one and only one element detuned in a controlled manner. For example, he adjusted each of the 675 'perfect' images to make the green channel one of three levels too high (33%, 66%, or 100% too high). Each of these detuned images is labeled with the error (e.g., OneLevelTooGreen001.png) so that a classification network can judge its success or failure in classifying the problem with the image. In total, he produced 24 sets of detuned images, each with a single problem area, creating a detuned database of 16,200 incorrect images. Though we did not need labeled images for the cGAN tests,

we utilized the same set of data in order to compare the quality of the output versus the classification network under the same training circumstances.

Images were reduced in dimension from 2K images (3840X2160 pixels) to a much smaller sized 456X256 pixels. Primarily this size reduction was needed to reduce system memory requirements and to substantially reduce the time it takes to process images in the network. In addition, color correction is generally focused on large-scale image problems, like the cast of a face, or the hue of the sky, so loss of smaller details is not much of an issue for images used to train color correction. As per the usual convention with CNNs, very small images are the expected inputs. More unusually, the original images are not (as per norms) square, but rather rectangular. For the classification network, which uses the VGG net as a start, images are expected to be 224X224X3 channels, so for that network, we 'squashed' images to that square aspect ratio (with 3 color channels). While this produces distorted images, it does not affect the aspects of the project we are interested in, and identification of problem classes was not an issue. As the classification network only recommends changes based on the error(s) in an image, distorting the input images has no discernable effect on the network's success. For the cGAN, input aspect ratio is not important, so we utilize the properly sized 456X256 pixel images to generate image pairs—uncorrected image on the left, and corrected image on the right—that are 912X256 pixels.

2.3 Experimental Setup

We simultaneously explored the two options for color grading. The first was to classify color correction errors via a classification network. The classification of one or more errors in the image could then be used by a color correction program to tweak parameters to correct for the noted problem(s). The second method was to use a generative network that can generate new

images that are able to fool a discriminator network as it compares the generator's output with the target (corrected) image. While both options use deep CNNs, the two paths are fundamentally different in their approach.

2.3.1 Classification Network

For our classification network, we utilized MatConvNet, an open source convolutional neural network construction system built to run within MATLAB (MatConvNet 2017), modifying the fast VGG classification network included in the package. As noted above, and shown in Figure 2.1, images were compressed in the horizontal dimension so that they filled a 224X224 square, which is the network's expected input dimensions. The images we used, which are in .png format, have values for each pixel that are by default doubles in UTF-8 encoding (even though they are 0-255 integers). As MatConvNet assumes every number in its data tensor is a single precision number, we had to convert the images (the .data tensor in the database) to single using the `single(imdb.images.data)` command in MATLAB. Though CNNs have traditionally been trained primarily to deal with the high frequency aspects of images, they worked very well, after retraining, for our focus on low frequency issues within the test images.

Figure 2.1: A sample input image with horizontal dimensions compressed to create a 224X224 square for the classification network

The classification network's goal is to identify what is wrong with the image (e.g., 33% too much orange, or black point set 66% too low) and return the result, allowing an eventual

automated plug-in extension—or a human user—to adjust settings in a color grading program. The primary advantage of the classification network is that it will 'do no harm.' In other words, given that it is a classification network, it will only tell a program (or user) which adjustments to make to fix a given problem (e.g., if the image is 66% too orange, the output would tell the plug-in to adjust orange color down by 66%). The primary disadvantage for the classification network is that it requires massive amounts of diverse, labeled data, which is not only time-consuming but a fundamentally challenging task. In our case, for example, we made 24 singular de-tuning adjustments to our 'perfect' images (black point 66% too high, etc.). This only accounts for one grading issue at a time, however. What happens when there are two issues simultaneously? Or when there is an unknown mix of issues? This problem can make generating properly labeled outputs for classification extremely challenging, as a color grader might make dozens of adjustments to get an image to look right to her. Thus without a large and varied amount of correctly labeled images to train on, the classifier might not generalize well.

To increase chances for a network that would work on a large class of images, we used a very low learning rate, and inserted up to three dropout layers (placed after the last three batch-normalization layers) with up to 80% dropouts on each of these layers to reduce the network's tendency to over-train rapidly. Though this slowed training down substantially, it proved to be ineffective at allowing the network to generalize as the results, discussed below, demonstrate.

2.3.2 Conditional Generative Adversarial Network

Our conditional generative adversarial network is a modification of the open source Pix2Pix cGAN that is built on the torch convolutional neural network framework (Torch 2017, Pix2Pix 2017). Modification of this network to strongly punish outlier pixels (e.g., introduced noise), to look at very large patches at a time, and to use temporal modifications described below, tuned

the network to train well with respect to our color grading issues. Prior to training, a script was run that paired detuned and tuned images, creating image pairs like the one shown in Figure 2.2. These image pairs were then fed to the cGAN for training.

Figure 2.2: An example image pair (uncorrected, very blue, image on the left; corrected, or target image on the right) that is fed into the conditional Generative Adversarial Network

The cGAN methodology changes input pixels to generate an entirely new output image that will (hopefully) fool an adversarial discriminator network into thinking its output is in fact the target image. The primary advantage of the cGAN is that it can utilize any set of corrected/uncorrected image pairs (of which there is a vast potential supply). An additional benefit of using a cGAN is that it provides a stand-alone solution to color grading: no other piece of software is needed to perform the color correction—as with the classification network—as the network generates corrected images itself. The primary concern with this network is that it will do damage to the image, reducing the quality or consistency of the output. An image, for example, might have its green cast adjusted properly by the cGAN, but the generator network might insert random noise into the image, or, worse, insert large-scale artifacts into the image. In these cases, the output images will be sub-optimal, likely to the extent that a viewer will notice the problems. Just as problematically, each image might be corrected well, but each image might be adjusted differently from others around it in a video sequence, thus causing images to flicker as they are presented at 24 or more frames per second.

For the cGAN, another significant concern is preserving high frequency detail while correcting large-scale, low frequency image issues. We tried several methods, some of which addressed the problem well (see Results, below). One of the reasons Pix2Pix was a good starting point for us is that the scripts utilize both patch and Euclidean error metrics, which account for feature matching and also per-pixel distance errors. Though it tends to produce blurrier images than the L1 (absolute distance) measure used by (Isola et al 2017), we used MSE (mean squared error) metrics to more drastically penalize rogue pixels in the output images. We also increased patch size and the metrics used for penalizing mismatched patches. Adjusting the network helped deal with unwanted, transient pixels and patches that would come across as flickers in a moving image, with the caveat that the individual images had softer edge detail due to these more draconian error metrics. As our ultimate goal is to produce color corrected image sequences (video), rogue or mismatched elements are unacceptable as they result in poor quality output, while slightly fuzzier images are not a noticeable issue in moving video, and there are as well ways to deal with softer images post hoc, as discussed below. In addition, we modified our cGAN to read in multiple frames of a video sequence at once by increasing the dimensionality of our tensor by one degree, accounting for a temporal dimension. Adding a fourth dimension to the tensor creates a X by Y by F by 3 (by batch size)[4] tensor, where the additional F dimension is the number of frames in a clip. While this addition increased memory requirements, training on sequential frames allowed the cGAN to learn to generate multiple frames that look alike, which is critical for making image sequences all look the same. Training individual images led to a flickering look, as each image was generated independently; sequencing images allowed the network to train to produce matched output for multiple images that were very nearly the same.

[4] X = image horizontal dimension, Y = image vertical dimension, F = number of frames in a clip, 3 = image color channels (RGB), and batch size = the number of images pulled into memory for simultaneous batch training.

2.3.3 General Setup

For both networks, we randomly selected approximately 2/3 of the 16,200 images for training, with about 1/6 for testing, and 1/6 held out for validation. While our concerns for each network were significant, they are, interestingly, complementary. While the classification network needs a labeled dataset and might not generalize as well, the cGAN is not much affected by these problems. On the other hand, where the cGAN might introduce noise or softness into the image, the classification network, as it only detects problems, cannot introduce any image degradation. For both networks, our interest was primarily in low frequency issues, like color casts or issues with contrast, as shown in Figure 2.3. Therefore we adjusted the parameters of each network to be more attuned to low frequency issues as opposed to the more usual concern researchers have with high frequency elements of images (e.g., feature detection).

Figure 2.3: Two source images showing the low frequency nature of color correction issues. The left image has its white point set 100% too low, while the right image has its contrast set 100% too high

2.4 Results

After adjustments and multiple training runs, both of our networks produced excellent results, solving the fundamental color correction task. Each network, however, exhibited some of the shortcomings predicted before trials began. Section 3.4.1 discusses results for the classification network while 3.4.2 discusses results for the conditional generative adversarial network.

2.4.1 Classification Network

For our classification network, optimal results for the training data were found very quickly, within 30 epochs of retraining the modified VGG-f network. As shown in Figure 3.4, convolution filter weights were indeed adjusted to deal more with low frequency, color-centric issues (note the blurring of filter outputs having to do with color, and with the more intense colors being output from many of the filters). In fact, in many cases classification confidence was at or nearly 100% for the correct problem classification, as shown in Figure 3.5. For the trained network, nearly all errors made were in neighboring classifications, creating "nearest neighbor" errors. For example, the network might predict that the white point was 66% too low, whereas the ground truth was that it was 33% too low. As this misclassification is qualitatively nearly correct, we factored these near misses into our results in addition to completely correct results, as shown in the middle column of Table 2.1. If one considers that the eventual outcome of this network is to recommend corrections—reduce the white point by 66%, say—then a mistake like this is not a great problem: reducing the black point by 33% rather than 66% is not going to make a drastic difference visually in the final image. Furthermore, on examining the probabilities for "nearest neighbor" mistakes, we found in every case that the correct classification also registers with very high probability. As an eventual correction (via software plug-in) would likely provide averaged rather than quantized corrections, for this example it might adjust the white point about 50% (the weighted average between the two), which would provide very acceptable results, especially as human qualitative viewing will be used to determine the quality of the output corrections. As Table 2.1 shows, error rates on validation data was exceptionally low for this network. The error rate was so low, in fact, that we feared the network was over-trained, as was borne out by subsequent experiments.

Table 2.1. Error rates on validation set for classification network, including correct and "nearest neighbor" errors in classification of image problems

	Correct Classification	Nearest Neighbor Error	Total
Number of Images	2,699/2746	46/2746	2745/2746
Percentage	98.3%	1.6%	100.0%

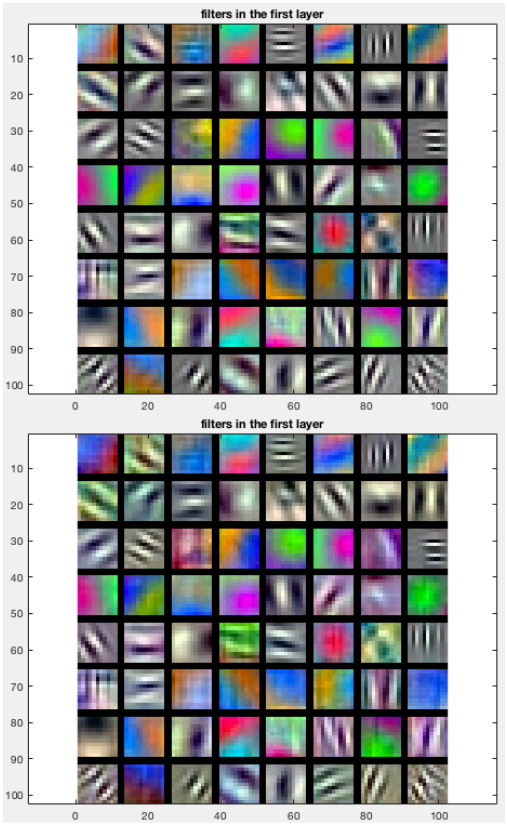

Figure 2.4: First layer convolution filters before retraining (top) and after 30 epochs of retraining (bottom), exhibiting a focus on lower frequency and color information when retrained

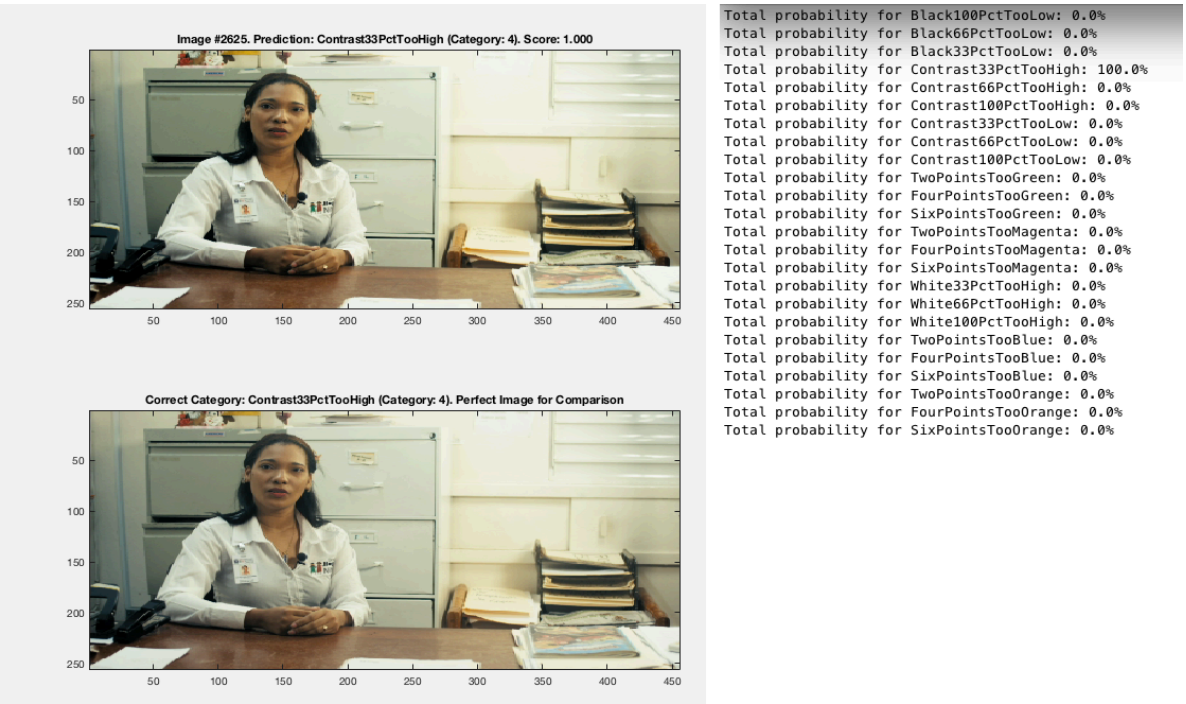

Figure 2.5: Correct classification that the source image's contrast is 33% too high. Note that the confidence in the result is 100%

To test our network under more real-world conditions, we input several images with two detuning issues, and a few images that were simply out of camera and not corrected. Unfortunately the network performed poorly on these. Attempting to run the network on images with two classes of problems at the same time, as well as on images with unspecified problems, demonstrated that the network failed to generalize properly, as shown in Figure 2.6. We thus adjusted our training methodology, most notably inserting three dropout layers after the last three batch normalization layers, with up to 80% dropouts. Though this slowed training down considerably it did not resolve the underlying issue of network generalization to other images.

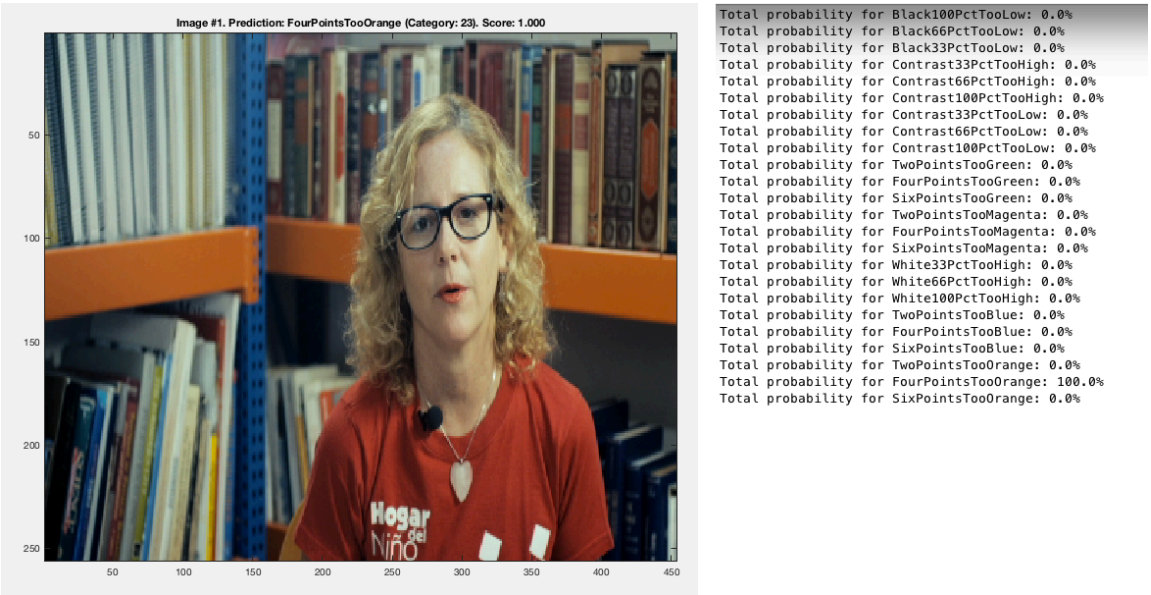

Figure 2.6: The network fails to generalize to images with multiple color correction issues. The network here predicts the image is too orange, with 100% confidence, while the true issues are too blue and too green

2.4.2 Conditional Generative Adversarial Network

Our cGAN with larger patches, MSE error metrics, and a high degree of weight given to the MSE portion of the loss function also produced very good, high quality results. As shown on the left and center images in Figure 2.7, the sample output image is nearly indistinguishable from the target image. The right-most image shows the results of subtracting the two images in Photoshop (via the difference layer mode—inverted to white for 0 difference, for clarity). That this image is nearly completely white, even for an exceptionally poor quality output image (based on output metrics), indicates that each pixel in the output image is extremely close to the value of the target image. Examining the image, approximately 74% of the pixels have integer values of 0, indicating that the pixel values in the two images are effectively identical. For better quality output images (the vast majority of outputs), the differences are much smaller.

Figure 2.7: cGAN output, left, compared to the target image, center. Right is the difference between the two images (via difference layer mode in Photoshop, inverted so that 0 difference is white instead of black). The nearly white result shows that most pixels have nearly the same value (the image on left is an unusually poor output, thus showing at least some difference between the two images)

As predicted, two significant issues pertain to the cGAN solution. First is that high frequency elements of the images are very slightly blurred, which was expected due to the highly weighted MSE factor in error accumulation, an error metric that tends to produce pixels more averaged over an image, especially in high frequency areas. One can see this effect in Figure 2.7, where edges are the primary area where there are differences between the left and center images, showing up as slightly colored pixels in the rightmost image. Though MSE did very well correcting for outlying pixels or groups of pixels, thus greatly reducing image noise, this comes at the cost of a slight blurring or softening of the image. Fortunately there is a simple solution to this problem: oversampling. This technique, which is used in many disciplines, including video games, blows up images beyond 100% before performing convolutional tasks on them. In our case, we doubled both the X and Y dimensions of our validation images (2X oversampling, which quadruples image size) before running the cGAN on them. After the network processed this larger image, producing a matching corrected one, we reduced the scale back to the original size. As the cGAN works well when applied to images larger than those it is trained on (see Isola 2017), the larger image size did not prove to be a problem for the network. Figure 2.8 shows a blown-up section of the same image run through the network at 100% per dimension versus

200% per dimension. While the differences between the images are subtle, there is a distinct sharpening of edge detail in the oversampled image. We could also, of course, train on oversampled data, though this requires more memory and time for the training.

The second problem area with the cGAN is more pernicious: as each image is run through the network individually, spurious pixels or patches can appear in one image that disappear (or move about the image) in the next. In addition, images can be corrected to different solutions when they are being created individually, thus producing images that have slightly different general characteristics (e.g., the color cast in one might be very slightly bluer than the color cast of another). When examining an individual image these rogue elements are generally slight variations, and thus are relatively invisible (or at the least inoffensive to the viewer). When viewed one after another in a moving image sequence, however, the changes between each image can produce a flickering appearance that is distracting.

Figure 2.8: The cGAN run on a standard sized image (100% in X and Y), on top, versus a 2X oversampled image (200% in X and Y), on bottom. A magnified portion of the image is shown here in order to reveal fine detail

We attempted two solutions for these inter-image problems, both of which worked well. Our first solution was to utilize post-hoc frame blending, a trick that has been used for years to good effect to match images. The left-hand side of Figure 2.9 shows a (greatly exaggerated) problem with

frames not matching, while the right-hand side demonstrates how frame blending reduces the differential changes between frames. Frame blending, as its name suggests, takes information from surrounding frames (backwards and forwards by some number of frames) and averages pixel values between them. While each frame becomes softer using this method, the moving video image, at 24 or 30 frames per second, is markedly improved, image flicker is reduced, and the individual image softness is not apparent.

Figure 2.9: Multi-frame differences produce a flickering effect, left (greatly exaggerated for clarity), while frame blending, right, smooths out frame-to-frame differences to produce a more pleasing sequence of images

Our other solution to the variability of sequential images was, as discussed above, to increase the dimensions of our data tensor to read in image sequences all at once, creating a temporal dimension. By altering the data tensor so that the cGAN trained on a four dimensional 'image'—X, Y, *image number*, color channel—it learned to generate sequences of images with little variation between them. Due to memory constraints we were limited to 18 images per sequence, but this number of frames was effective at reducing variation between frames to a very small amount. One interesting discovery when training on four dimensional images was that flipping the horizontal dimension of random images within the sequence actually worked better than keeping them all in their original horizontal configuration. As far as we know there is not a rigorous explanation why flipping images during training helps, but practice is to flip images randomly as it empirically improves results (see Pix2Pix 2107 for a discussion). Figure 2.10

shows a portion of an image sequence (with random flipping), indicating that this solution creates consistent output images over a sequence.

Figure 2.10: Training the cGAN on image sequences as a group produces a more consistent look. Left is the uncorrected input, middle the cGAN output, and right, the corrected target image. Note the random image flipping in the sequence

Very importantly, the cGAN generalizes well. Given image types the network has not trained on at all, as in Figure 2.11, the network produces reasonable quality results, indicating that even with a small and specific data set to train on (i.e., talking heads video sequences), it already can generalize to a larger class of uncorrected images. With a larger, more diverse training set, the quality of output should improve even more.

Figure 2.11: Given totally new types of input images, the cGAN produces good quality results. Original uncorrected images are on the left, while corrected output images are on the right

2.5 Discussion

Both the classification and the conditional generative adversarial neural networks produce very high quality results. Comparatively speaking, the classification system's main shortcoming—its inability to generalize well beyond a single-problem, labeled data set—is likely to be a more significant issue than the two problems the cGAN has—softer details and rogue, changing elements in succeeding images (creating image flicker).

The only really effective way to create a more robust classification network is to accrue, and more problematically, properly label a large database of images. Labeling is not only time consuming but also highly challenging, as any number of subtle corrections can be performed by a color grader while working on an image. Effectively notating the range of input image problems being corrected for by the colorist given a real-world image is something that could perhaps be solved by keystroke recording software. The number of classification categories, however, would then become problematic. A colorist might, say, make 20 changes to one image sequence, and 20 changes to another image, but these changes will almost certainly not be identical, and thus each set of changes needs to be its own classification category. Given that each change can at the least go from 1% change to 100% change (and very possibly more than 100%), the set of classification categories, even assuming each category only accounted for one integer percentage point at a time, could be 20 • 100, or 20,000 categories for only these 20

change categories. Thus the number of possible categories would grow into the tens of thousands, massively increasing training time and likely reducing the effectiveness of the network as a whole, as it would have to discriminate between very subtly differing categories.

The cGAN's issues, on the other hand, have already been partially addressed even with the limited initial data set used. Edge softening is already effectively taken care of, as oversampling (followed by image reduction after processing) has been added to our pipeline, and works very efficiently to sharpen edges and other high frequency elements in the images. Working with images larger than those trained on is also not a problem, as there have been few issues noted in our tests. Furthermore, with more time and resources, training can easily be done on larger images, almost certainly improving results further.

Image-to-image variance is the outstanding issue with the cGAN. We tried two solutions to this problem, each of which had a positive effect. Our first solution was to post-process the image sequences using frame blending. This solution works well, but there can still remain subtle flickers, and unfortunately the individual images are degraded, as they become somewhat smoother and blurrier, which while generally undetectable for a viewer is nonetheless a reduction of overall quality and fidelity to the source images. Our second solution was to add a temporal dimension to our image data tensor. This addition, while increasing memory load on the CPU/GPU system, allows for image sequences to be corrected as a unit, creating a correction that accounts for a sequence of very similar images (as they are sub-second frames in a video clip) rather than to individual frames. This solution drastically reduced inter-frame differences in the video clips we used to validate our results. One other potential solution is to modify our code to include temporal convolution as in (Ji et al 2013), though it is not obvious that this will

produce better results than the image sequence modification we have implemented, as a combination of frame blending and image sequence training created nearly ideal output.

Both our classification network and our conditional generative adversarial network were trained to produce high quality output from the data we gave them. The classification network could easily identify (classify) problem areas in an image that had a single detuning error. The cGAN produced images that are nearly indistinguishable from the target output images. Our opinion is that between the two networks, the cGAN system is more suited to further research, as collecting unlabeled image pairs is relatively easy and straightforward, and as the issues still outstanding are partially solved, and thus more tractable.

Though color correction is considered an aesthetic task, both of our neural networks learned the basics of the task using just 16,200 training images. Furthermore, as the intended output of these networks is to create qualitatively correct, rather than quantitatively correct, images—in other words, to please human judges rather than achieve a certain numerical metric—the fact that they can produce images that are considered correct by a domain expert is the most important factor in their utility. We believe that with further training on larger, more diverse data sets, our cGAN network in particular can provide a practical solution to a complex, time-consuming, artistic task with which every film/video producer has to contend.

CHAPTER 3

IMAGE RETRIEVAL VIA CLASS-BASED HISTOGRAMS[5]

3.1 Introduction

In the last few years, users' desire to find more images like the one they are currently viewing has increased dramatically. From personal photo libraries to individual and business searches, vast numbers of image consumers are interested in finding images that qualitatively match the one they are viewing at the moment. As the quantity of stored images has expanded to a number far beyond what any team of humans could examine, classify, and catalogue, we have turned to machines running Artificial Intelligence searches to do the work for us.

The industry terms for recovering images that are visually and semantically similar to the search image are Content-Based Image Recall (CBIR) or Image-Based Recall (IBR). At first, IBR required a great deal of data processing and tuning in order to get generally sub-par results (i.e., much worse than a human would do performing the same task). Recently, however, AI search methods have become so sophisticated and robust that raw images can be submitted to an IBR engine and quality results returned in a very brief time. The major IBR breakthrough in the past few years has been the use of deep convolutional neural networks. Large search engines like Google, shown in Figure 3.1, Duck-Duck-Go, and others, present images the search engine determines to be relevant to the one a user has selected.

[5] This chapter is an extended version of the following published paper: (Kundert-Gibbs 2017).

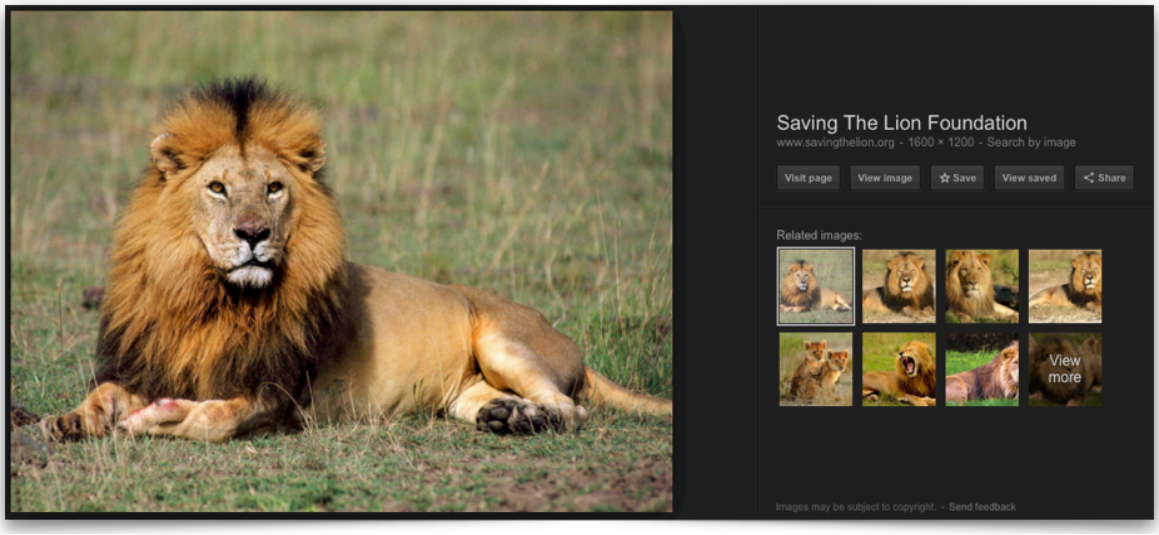

Figure 3.1: Google image search presenting images (right) relevant to a query image (left)

Even with major advances in IBR, the area is an ongoing topic of research as results are not consistently appropriate.

We propose a new system that can outperform publically available IBR packages on a reasonable size database of images. Our system utilizes a class histogram approach (described in section 2.3.2) to compare a query image to scores from an image database, producing quality results rapidly. Though evaluating IBR can prove challenging as the results are generally qualitative (i.e., does image 1 look like image 2?), we can make some quantitative assessment as well, using the error factor generated by code to examine how closely these systems think returned images are to the search image. By comparing our un-retrained and retrained networks to two off-the-shelf IBR solutions, we find that our system works better than the publically available systems, and that further training increases the accuracy of our method. Though the two IBR packages ours is compared to are not nearly as robust as commercial IBR packages (like Google's image search), these commercial packages are proprietary, so it is difficult to compare results. Though

we do not have access to these state-of-the-art packages, we nonetheless can see that our system outperforms what was publically available in 2017, when the research was conducted.

3.2 Current Research In Image-Based Recall

Substantial work has been done on the topic of IBR for more than two decades. Most of the traditional methods (Bishop 1995, Kearns 1994, Mitchell 1997, Zhou 2001, Zhou 2003) require a large number of training instances. These also require seeding a query with "good" examples (Jones 1997, Porkaew 1999, Wu 2000, Ortega-Binderberger 2003). Until recently most training sets were relatively small, thus IBR engines did not have much to work with. Even with the advent of large image databases like Image-net, and new techniques like Support Vector Machines (Tong 2001, Hearst 1998) and active learning (Cohn 1996), results have been only marginal. Some approaches making use of ensemble learning such as bagging (Breiman 1996), arcing (Brieman 1998), and boosting (Freund 1995), have improved results by combining other methods, including decision trees (Quinlan 1986, Quinlan 1996) and neural networks (Nigrin 1993). These ensemble schemes have been successful at improving classification accuracy through bias or variance reduction, but they do not help reduce the number of samples and the time required to learn a query concept. In fact, most ensemble schemes actually increase learning time because they introduce learning redundancy in order to improve prediction accuracy (Dietterich 1995, Grove 1998, Moreira 1998).

An approach based on Support Vector Machines (SVMs) is proposed in (Tong 2001), but this approach requires seeds to start, which is not practically feasible, especially for large database queries. An advanced approach using active learning was proposed by (Chatterjee 2015). This approach suffers from the limitation that the model used for retrieval has to train on annotated

images before it can even be deployed for testing, which is at best extremely time consuming and at worst practically impossible due to data availability and hardware limitations.

Over the past decades, a variety of low-level feature descriptors have been proposed for image representation ranging from global features, such as color features (Jain 1996), edge features (Jain 1996), texture features (Manjunath 1996) GIST (Oliva 2001, Oliva 2002) and CENTRIST (Wu 2011), and also local feature representations, such as the bag-of-words (BoW) (Sivic 2005, Yang 2007, Wu 2010, Wu 2011) models using local feature descriptors (e.g. SIFT (Lowe 1999) and SURF (Bay 2006)). Of note, one of our baseline packages, CBIR (see below) utilizes some of these feature representation modes. Conventional IBR approaches usually choose rigid distance functions on some extracted low-level features for their similarity search mode, such as Euclidean distance. However, a fixed rigid similarity/distance function may not be optimal for complex visual image retrieval tasks. As a result recently there has been a surge of research into designing various distance/similarity measures on low-level features by exploring machine learning techniques (Wu 2011, Norouzi 2012, Chechik 2010, Chang 2007, Salakhutdinov 2009a). Distance metric learning for image retrieval has been extensively studied (Chang 2007, Domeniconi 2002, Bar-Hillel 2003, Weinberger 2005, Lee 2008, Guillaumin 2009, Wang 2013b, Mian 2013, Wang 2013a). Some of these methods focus on hashing or compact codes (Salakhutdinov 2009b, Norouzi 2012, Jgou 2012, Zhang, L. 2014, Zhang, Y. 2014). Jgou (2012), for instance, adopted the fisher kernel to aggregate local descriptors and adopted a joint dimension reduction in order to reduce an image to a few dozen bytes while preserving highly accurate feature elements.

Another way to enhance the feature representation is distance metric learning (DML). The key idea of DML is to learn an optimal metric that minimizes the distance between similar images

and simultaneously maximizes the distance between dissimilar images. Distance metric learning for image retrieval has been studied in both the machine learning and multimedia retrieval communities (Domeniconi 2002, Bar-Hillel 2003, Weinberger 2005, Lee 2008, Guillaumin 2009, Wang 2013a, Mian 2013, Wang 2013b). In some instances like (Weinberger 2005) class labels are used to train DML. Distance metric learning techniques are typically categorized into two groups: the global supervised approaches that learn a metric on a global setting by satisfying all the constraints simultaneously (Bar-Hillel 2003, Hoi 2006), and local supervised approaches that learn a metric using a patchwork technique, only satisfying the given local constraints from neighboring information (Weinberger 2005, Domeniconi 2002). Most DML studies employ batch learning methods which assume the whole collection of training data be given before the learning task and then trains a model from scratch. Unlike the batch learning methods, online DML algorithms have been studied recently as a means to handle very large image data sets by onboarding images as they arrive, thus doing just-in-time training (Jain 2008, Jin 2009).

Recent advances in deep learning have created several high quality IBR techniques. Deep Learning lies at the intersection of several research areas, including neural networks, graphical modeling, optimization, pattern recognition, and signal processing. Deep learning has a long history, and its basic concept originated from artificial neural network research, which stretches back into the 1950s. Back-propagation, popularized in the 1980s (see for example Yoon 1990), is now widely used for training the weights of these networks. For example, (LeCun 1998) successfully adopts the deep supervised back-propagation convolutional network for digit recognition. Recently, deep learning has become a hot research topic in both computer vision and machine learning, where deep learning techniques achieve state of the art performance for various tasks. The deep convolutional neural networks (CNNs) proposed in (Krizhevsky 2012)

received first place in the 2012 image classification task, ILSVRC-2012, proving the worth of this rejuvenated network architecture.

Over the past several years, a rich family of deep learning techniques has been applied to the field of computer vision and machine learning. Just a few examples are Deep Belief Networks (Hinton 2006), Boltzmann Machines (Ackley 1985), Restricted Boltzmann Machines (Salakhutdinov 2007), Deep Boltzmann Machines (Salakhutdinov 2009b), and Deep Neural Networks (Hinton 2012, Krafka 2015).

For our method, we make use of a pre-trained VGG- 16 model which was released by the Visual Graphics Group of Oxford University. A schematic of a deep convolutional neural network is shown in Figure 1.4, and repeated here as Figure 3.2.[6]

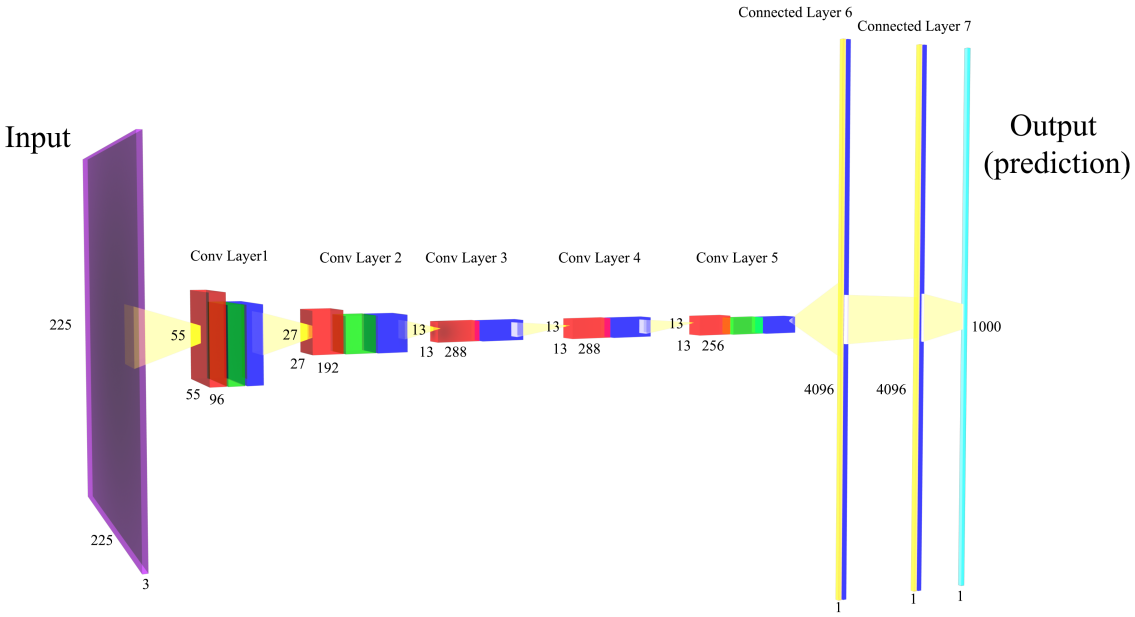

Figure 3.2: A schematic of the architecture of a deep convolutional neural network, or CNN

Some very recent work has been done on image classification using histograms in the medical field (Rahman 2017) and texture classification (Kulkarni 2018). While this work is not directly

[6] See Section 1.3.2, on page 9, for a discussion of the architecture of CNNs.

related to our work here, it is interesting that other researchers have begun to make use of histograms in combination with neural networks.

3.3 IBR Packages

While a number of IBR packages exist, we found two packages based on MATLAB that are good experimental candidates because they are publically available, building on the MATLAB foundation, and are consistent in their underpinnings, using scripts that are open to examination. These two IBR implementations serve to provide baseline results for comparison with our IBR method, which is also implemented in MATLAB. While neither off-the-shelf IBR package, cbires and CBIR, provides state-of-the-art results, the best results today are from corporations like Google, and thus their code is proprietary, so it is difficult to impossible to compare our results with theirs. While neither cbires nor CBIR provide stellar results—neither is up to the standards of professional IBR code being used commercially—they do provide a baseline for comparison, as we can compare results with the same database and with control over the experimentation. With two open source packages to compare with our method, all written as scripts in MATLAB, we can fully explore the effectiveness of our IBR strategy versus these others in a controlled environment.

3.3.1 Previously Available IBR Packages

The first package examined is cbires, developed primarily by Joani Mitro. cbires uses either k-nearest-neighbors (knn) or Support Vector Machines (SVM) plus feature extraction to perform IBR (Mitro 2016, cbires 2017). As described in (cbires 2017), a good deal of data pre-processing is performed by cbires before the IBR techniques are used. An HSV histogram, 4X4X4 auto-correlograms, RGB mean and standard deviation and Gabor wavelet coefficients are all calculated for each image. Each image is then decomposed in a 3-level decomposition, and all

resulting values are placed together in a vector that describes each image. An image database can be created from any set of sample images.

Once images are pre-processed, cbires can use either knn or SVM to recall images similar to ones in the pre-created database. All essential controls are available via a GUI shown in Figure 3.3. One either loads or creates an image database first; then one loads a test image (which, unfortunately, must be part of the training set!); then one chooses either “query” (knn) or “Support Vector Machine” as the IBR engine.

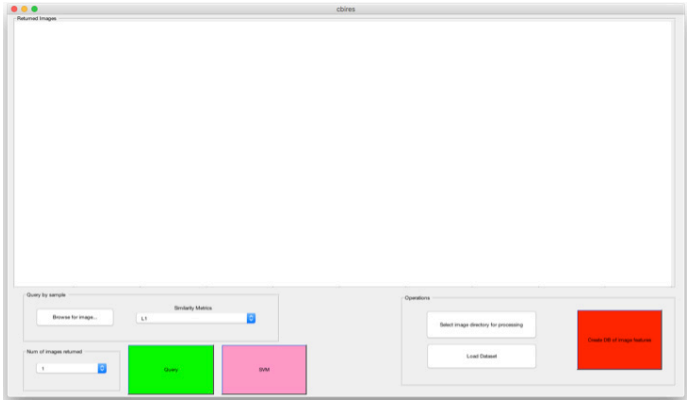

Figure 3.3: The Graphical Interface for the cbires package

The second package, CBIR, was developed by Amine Ben Khalifa and Faezeh Tafazzoli (Khalifa 2013, CBIR 2017). CBIR utilizes feature extraction, which can either be done locally or globally. Both color and texture features can be extracted (with much more user control than the cbires package) either globally or locally, and different “distance” measures can be invoked to compare images, including Euclidean, Quadratic, and Chi Squared, to name a few. The user can also choose how to weight colors versus features, as well as other options, like the number of color bins, and so on. CBIR functions much like cbires in that it uses feature extraction first, and then calculates a “distance” between sample image and images within its database. With so many options, as shown in Figure 3.4, trying to achieve the ideal settings was not trivial for CBIR.

Also of note, CBIR does a great deal of calculation when sample images are queried, so it is runs the slowest of all tested methods during an IBR query.

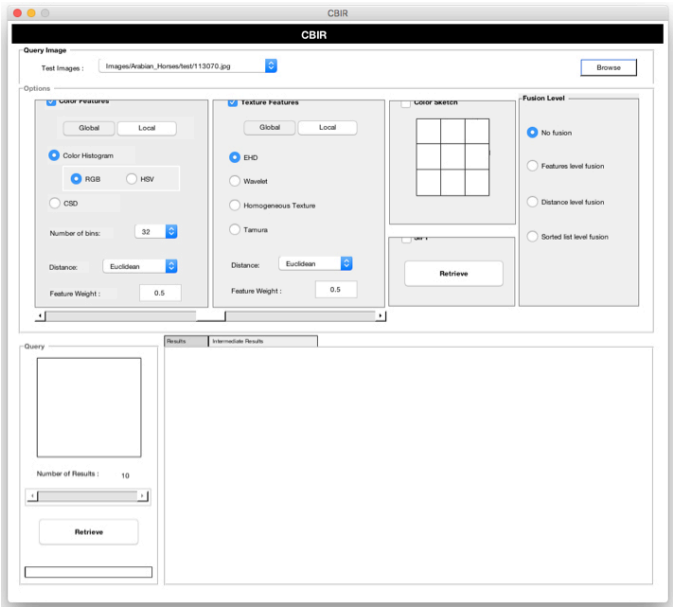

Figure 3.4: The Graphical Interface for the CBIR package

3.3.2 Class-Based Histogram IBR

The method we have developed operates differently than the two baseline IBR packages described above. As the package is in a developmental stage, there is currently no GUI for it. While work has been done on per-image histogram comparisons to classify content (see for example, Swain 1992 and Wichmann 2002), our system differs from these in its use of a meta-histogram: the histograms being compared are not of the images directly but of their classification probabilities. This system, which we have termed Class-Based Histogram, or CBH-IBR, uses a pre-trained deep learning convolutional neural network (DLCNN)—in this case trained on the Image-net database (image-net.org 2018)—as the basis for image recall. A DLCNN uses a deep (21 layers in this case) convolutional neural network to classify images. In our case we modify a network trained via matconvnet (Vedaldi 2013)—a script package for

MATLAB that is specifically designed to create and train convolutional neural networks—that is set up to classify the 1,000 categories of images that Image-net contains. While this network, imagenet-vgg-f.mat, which comes included with the matconvnet download, is intended for use classifying a single output class (e.g., "this is a picture of a pomeranian"), we note that the final layer (a fully connected softmax probability layer) produces a 1,000 element vector that contains a probability between 0.0 and 1.0 for each of the classes. We exploit this fact by running a MATLAB script that records the full 1,000 element vector for each image in a resource database (from which images are pulled to match the query image). These vectors create a histogram of each of the 1,000 possible classes. When a query image is submitted via another script, its class vector is calculated and then compared via SSE to each of the other images, as shown in the following formula:

$$S_{best} = \min\left(\sum_{i=1}^{\check{} } \sum_{j=1} (q, q = queryim, b = baseimc \right.$$

In other English, the query image's class prediction vector (1,000 elements) is subtracted term-by-term from the image class vector (also 1,000 elements) of each database image, in turn, and each of these values is squared, producing a positive number that is added to the other 999 items to produce, in the end, a sum of the squares of the differences between the query image's histogram and a given image in the database. The size of the number is considered a proxy for the "closeness" of a given database image to the query image: a smaller number means the two images are more closely related in a quasi-semantic manner (they look similar to each other). The n images with the closest matches (smallest differential SSE, or smallest distance from the query image) in the resource database are chosen and displayed, as is the error between the query and resource images.

Even for images that do not contain one of the 1,000 image-net classes, each histogram turns out to be distinct—a kind of fingerprint for each image—and therefore can be used to retrieve similar images. Figure 3.5 shows a graph of the histogram of three images: two are very similar (a husky and a beagle—both dogs) while one is quite different (a rock climber). Note that rock climber is not a valid image-net classification category, and thus the histogram has no outstanding peak. This image can still be used to retrieve similar images, however: even though there is no distinct probable output class for the image of the climber, the histogram is distinct from others and similar images produce similar histograms. The fact that images beyond the 1,000 classes defined in the image-net set can be queried and discovered is exceptionally useful as it means this method can recall images it was not trained to recognize at all. One can see from Figure 3.5 that the similar images have very similar histograms, while the different object has a very different histogram. After some experimentation, we decided to discard all probabilities below 0.01 when comparing images: these tiny probabilities add noise to the comparison and reduced the effectiveness of the engine. We have used our CBH method to test and recall many query images that are not classified within the 1,000 image-net classes (along with many images that are from image-net classes). While these images would not produce viable classifications via the network (as there is no class for them to find in the 1,000 classes), they nonetheless produce good results for IBR.

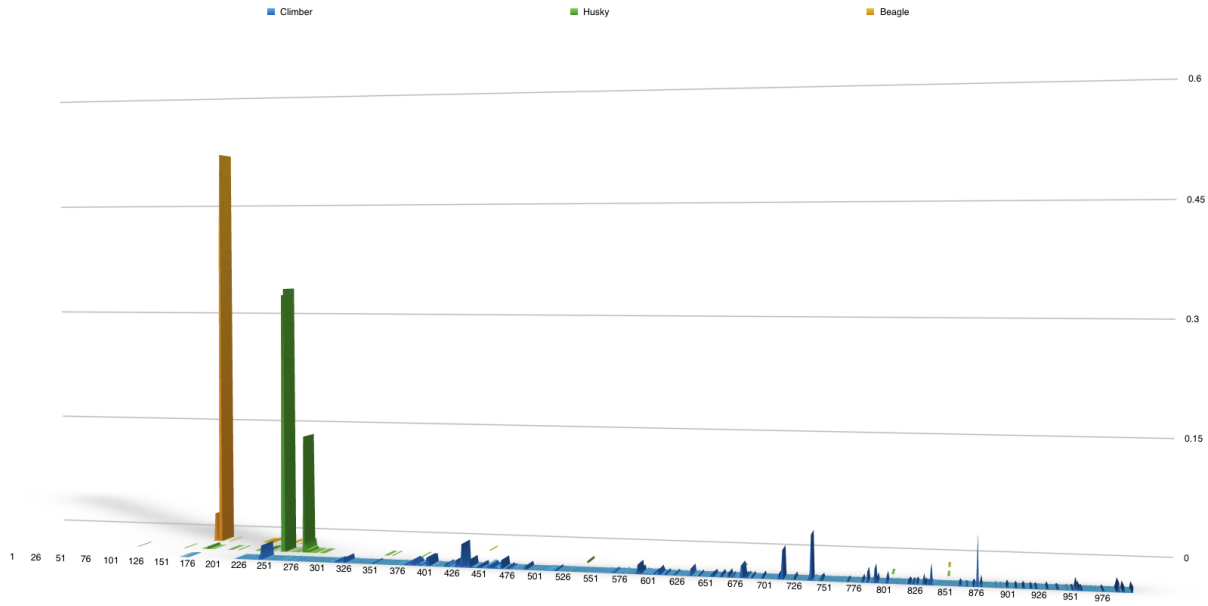

Figure 3.5: The histogram output of two similar images (a husky and a beagle) and one dissimilar image (a rock climber). Note that the similar images have similar distinctly spiked histograms, identifying a class within the image-net databas, while the dissimilar image—which has no class in image-net—has a very different histogram that is much more spread out across classes.

3.4 Experimental Setup and Methodology

How to compare different IBR techniques is not entirely obvious: as IBR produces qualitative results (i.e., how similar is one image to another?) human judgment is required to determine whether an image is similar or not. Additionally, there is no consistent error measure between techniques. For example, our CBH-IBR method uses the Sum of Squared Errors (SSE) of the histograms of image classification to compute error, while cbires uses Euclidean distance of feature vectors, and CBIR uses one of several methods to compute error. Thus there is no simple way to compare error measures to determine if one search engine does better than another. We therefore turned to a counting method—precision—to determine the quality of results: we counted images that are categorically classified as the same as the query image (class matching); as we are looking at qualitative matches, we also counted images that are "pretty

close" to the query image even though they are not classified the same (e.g., a sunset query image might be a sunset over a beach and ocean, and a matching image might be of a beach and ocean during the day, or a sunset over the desert, either of which is "pretty close" to the query image); finally we counted images that are "not at all close," i.e., both categorically and qualitatively incorrect. From these relatively straightforward metrics we calculate the precision of our results, either using only categorically matched images, or both matches and "pretty close" images. As precision is $\frac{TruePositives}{TruePositives + FalseP}$, for the categorical (exact) matches, any returned image that does not match the category of the query image (e.g., 'bear') is counted as a false positive. For the looser, qualitative matches, we (using human judgement) place "pretty close" images into the True Positives category as well. In the results section, we provide the precision measure for both the categorical match results and the categorical match + "pretty close" results.

We selected two image sets, the Caltech 256 data set (vision.caltech.edu 2018) and the one included with the CBIR package (Khalifa 2013), and combined them into an image database of 29,970 images that fall within 271 classes (many of which are not Image-net classes). These images contain between 80 and 200 of each image class/descriptor (e.g., sailboat, horses, bear, car).[7] We then selected 50 images from google.com and duckduckgo.com as test query images (image descriptions are listed in Appendix A). The images are chosen to be reasonable query images, given the source image database; in other words, images that are similar to a large number of images (at least one class of 80+) within the source images.[8] These images are

[7] As the Caltech data set contains many classes with more than 200 images, while others have as few as 80, we removed any images beyond 200 for a class to reduce class imbalance.

[8] As a test, we tried running our engines with query images that had no close relatives in the source images. Our CBH-IBR found images that had a qualitatively similar look, even though there were no possible exact matches.

isolated from the query database and any training work, so that they remain completely outside the world that the IBR packages had access to for training or querying.[9] For each engine, after adjusting to find optimum settings, we run a query for each of our 50 test images and request 20 similar images be output. For each of the 50 searches (requesting 20 images similar to the query image for each search) we count up the number of correct images (matched categories), the number of partially correct, and the number of incorrect results, and record them in a spreadsheet. Precision measures are computed for each image query as well as a single precision result for the entire 50 image query set for each query technique, shown in Table 3.1.

In our tests, our pretrained Image-net network works very well, but still has room for improvement. We thus tried numerous methods to retrain/refine the network, including retraining via softmax log loss, top k error, mshinge, and our own modified version of softmax log loss that accounted for the top 10 matches (each altered to return a softmax log loss). While our hope was to find one method that outperformed the original network in all cases, this did not occur. We thus created a voting method that utilizes the best three retraining methods—the original network, the network retrained with softmax log, and the network retrained via minimizing sum of squared errors on CBH-IBR—creating a results vector of all three methods combined. We sort this new, combined vector (3 times the length of each return vector, or 60 values in this case) and take the top 20 results. While a few results are actually worse, most are the same or improved, so this method produces the best overall precision, as presented in Table 3.1.

This "graceful failure" result is not easily quantifiable, but seems similar to the way a human would select a close image to a query one if no exact matches were available.

[9] cbires requires the query image be in the image database, so we had to place the query image in the database before performing cbires searches.

Table 3.1: Precision measures for each of the four IBR methods tested (best results in bold).

IBR Method	Precision (match)	Precision (match + partial)
cbires	0.138	0.313
CBIR	0.125	0.332
CBH	0.778	0.896
CBH - Retrained	**0.866**	**0.959**

3.5 Results

While the results in the right-hand column of Table 3.1 are somewhat qualitative, as we have to use human judgment to determine how close the more qualitative results are, the middle column, as it relates to categorical matches, provides a more rigorous comparison. In either case, however, it is easy to see that our CBH method provides far better results than either of the two packages to which it is compared. As the images below demonstrate, our IBR engine works substantially better than the baseline packages using the same source image database and the same query images. While both cbires and CBIR work fairly well on the data set on which they were tested, these two data sets are much smaller and less varied than the one on which we ran our tests. It was disappointing, in fact, how poorly both cbires and CBIR did on our data set.

As IBR is visually based, we first present some representative results we have achieved for the different IBR engines. The cbires recall engine performs poorly, as evidenced both by its total precision score and by observing results. We attempted to improve the results, but there are very few parameters that can be adjusted via its GUI. We tried both knn and SVM methods, and found them about the same. Our results are for the knn method. As Table 3.1 and Figure 3.6 indicate, the results of cbires are inadequate, with similar images only appearing in a few result

images, and those similar images are not even the ones calculated to have the smallest error (which are near the top, next to the query image, top-left).

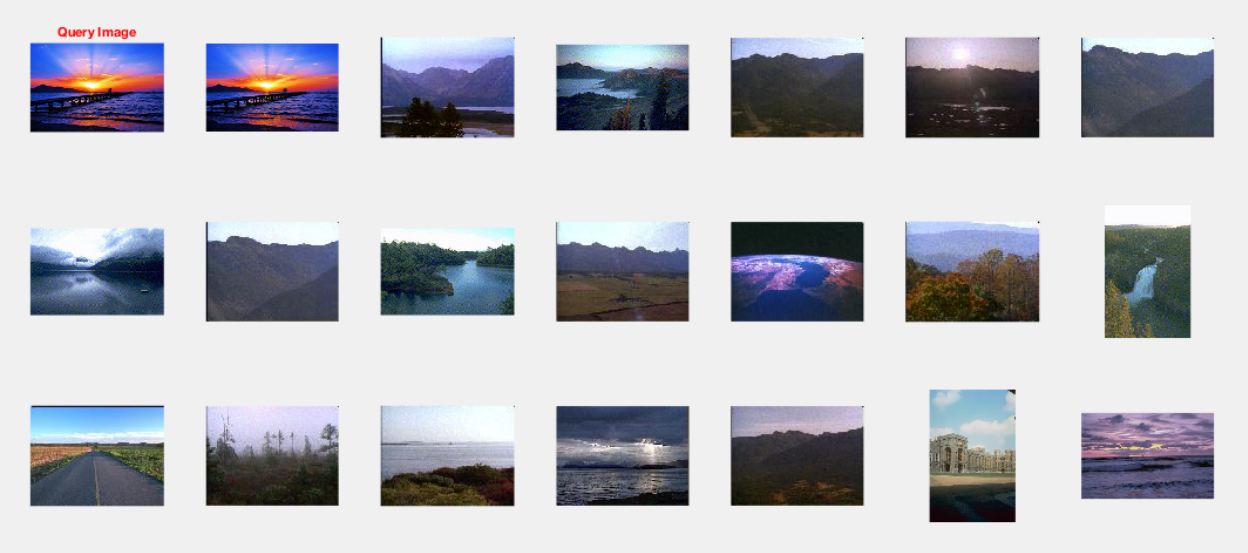

Figure 3.6: Using cbires to query a sunset image. 19 images (plus the original) are presented

CBIR performed around the same as cbires after some tweaking. Even at the best settings we could find, however, the image query results are certainly less than adequate, as indicated in Table 2.1 and Figure 3.7.

Figure 3.7: Using CBIR to query a bear image. 20 images (plus the original) are presented

As opposed to the two baseline methods, CBH-IBR produces high quality results, both visually and via the precision measure. Without retraining, the only tuning adjustment for this system is whether to ignore small values in the histogram vector, and what the threshold should be for ignoring small values. Empirically we determined that a value of 0.01 (or 1%) works the best. This setting ignores the noise of any very small probabilities, improving results dramatically. Interestingly, a large value for the threshold percentage reduces the quality of the results, indicating that categories of classification with smaller values significantly improve the engine's ability to find similar images. Figures 3.8 and 3.9 show two excellent results, while Figure 3.10 shows one that is not wholly adequate.

Figure 3.8: Using CBH-IBR to query a classic car image. Only 10 images (plus the query image, repeated for visual balance) are shown, for space purposes

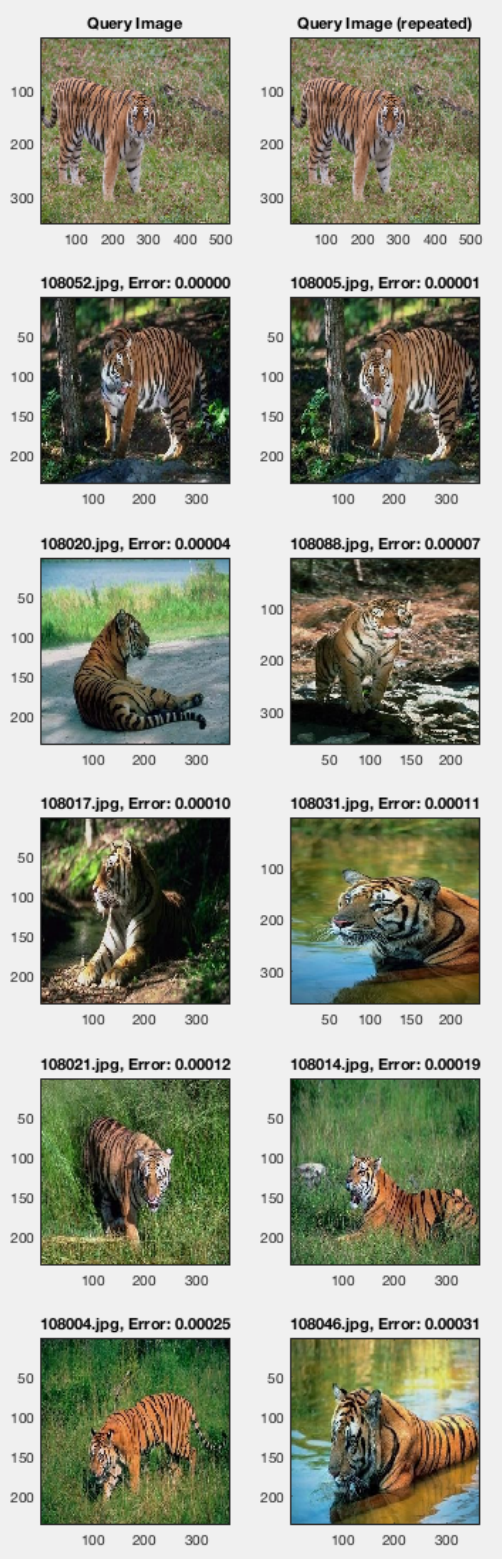

Figure 3.9: Using CBH-IBR to query a tiger image. Only 10 images (plus the query image, repeated for visual balance) are shown, for space purposes

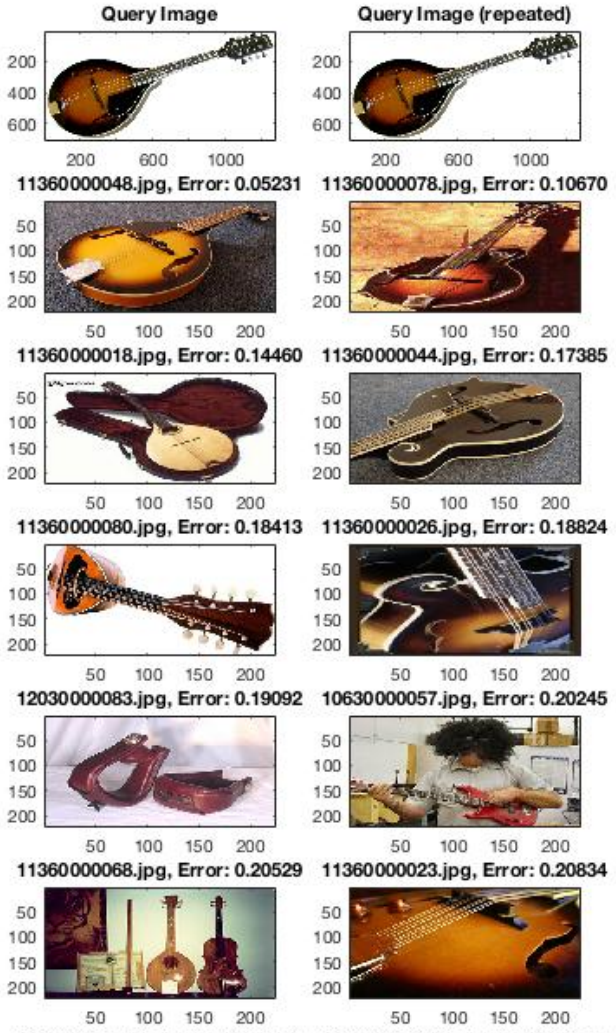

Figure 3.10: Using CBH-IBR to query an image of a mandolin. Only 10 images (plus the query image, repeated for visual balance) are shown, for space purposes

Retraining the CBH-IBR method involves assigning classes to each image in the database, and using one of the above mentioned prediction/loss algorithms to determine the quality of the result. Retraining improves results in many cases, but also creates instances where the results are worse than the original. Thus we have stacked the three best methods—the original network, one

retrained via softmax log loss, and one via a custom histogram/class method[10]—and use lowest error scores from amongst the three methods to generate our 20 results. Figure 3.11 shows the results of the Mandolin query from Figure 2.10 after retraining. Note that the two images in the second to last row are now images of mandolins, not incorrect results. Though this stacked network method works the best of any we tested, it still produces results that are less than perfect for some query images, as shown in Figure 3.12.

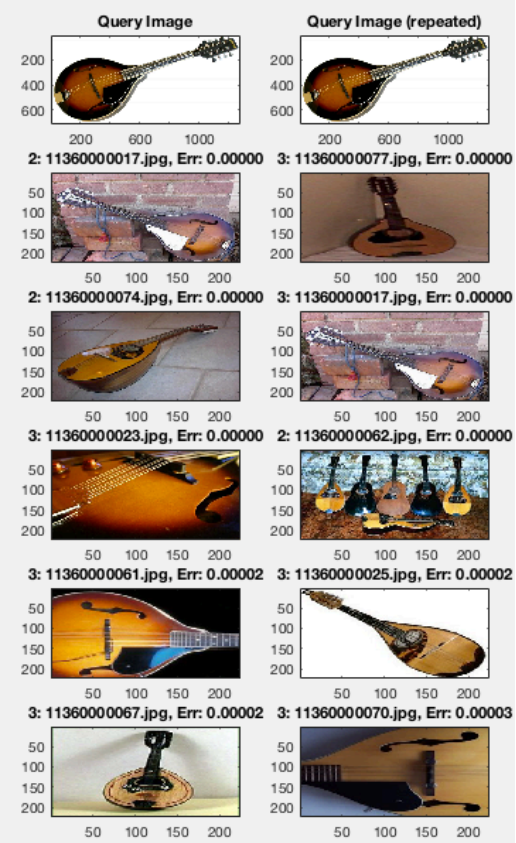

Figure 3.11: Using 3 CBH methods to query the same image of a mandolin as in Figure 10. Only 10 images (plus the query image, repeated for visual balance) are shown, for space purposes

[10] Our method minimizes the derivative of the sum of squared errors between class histograms (term 1) added to class error, or softmax loss (term 2). The -1.5 term is an empirically determined weighting between the two terms.

$$S_{derivative} = dzdy * \left| -1.5 * \left[\sum_{i=1}^{images} \sum_{j=1}^{classes} 2(q_j - b_{ij})^2 \right] - \sum_{i=1}^{imc} \right|$$

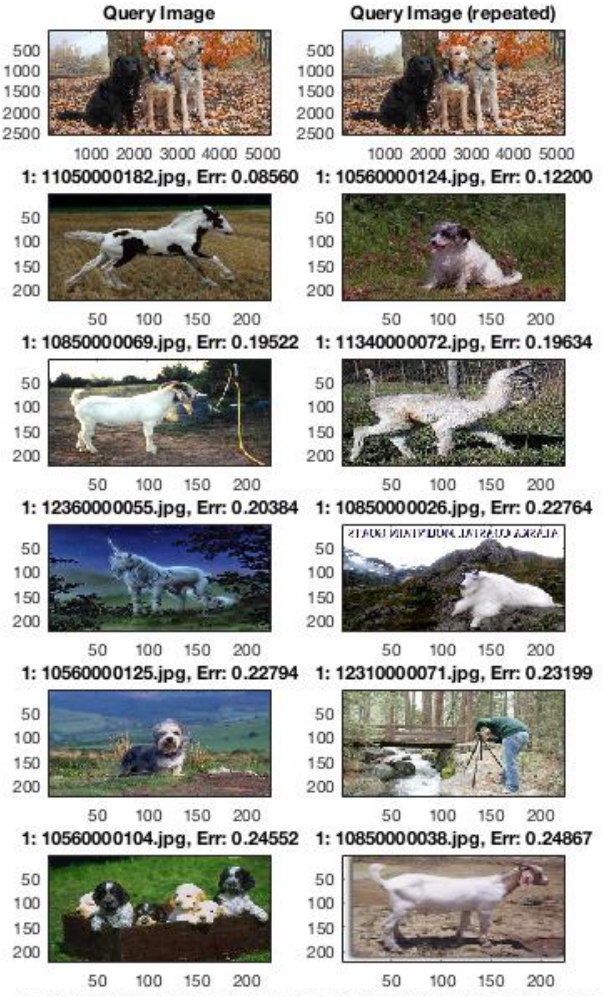

Figure 3.12: Using 3 CBH methods to query an image of 3 dogs. Only 10 images (plus the query image, repeated for visual balance) are shown, for space purposes

From visual examination, we produce precision measures for each search method, and for both categorical matches and matches + partially matched results. Table 3.1, above, shows the results of these more quantitative measures and the numbers parallel our visual observations.

3.6 Discussion

Our CBH-IBR method produces substantially better results than the baseline CBIR methods we tested. These results indicate that comparing class probability histograms produces high quality results without needing much data preprocessing (beyond adjusting images by subtracting the overall image database mean, which is standard practice). In addition, retraining

the network improves results, though sometimes at the cost of exact matches, as, shown in Figure 3.14. While images on the right of Figure 3.13 are correct, as they match the class bear, they are visually less correlated than the original network's results, as they output more brown bears. Our retraining method thus needs more granular classes on which to train: with sub classes for various bears, the degraded results would be eliminated. We also note that retraining based on individual classes is likely not the best way to improve performance. Our method allows the network to discover a web of probabilities associating images. Retraining the network to recognize a single class as correct, while improving class-based results, does not necessarily improve the visual quality of the results. We submit that a substantially different training method needs to be developed to refine the training of these networks. This could simply involve further training an original network like the image-net one with more images and more classes, but something more clever that can bootstrap the visual quality of results might work even better. Our custom prediction/loss method, which involves minimizing the sum of squared errors between the class histograms of the query image and the training images, works well only if we add to this the standard softmax loss method for classes. While results of this custom loss method are encouraging, research into improving this method is ongoing.

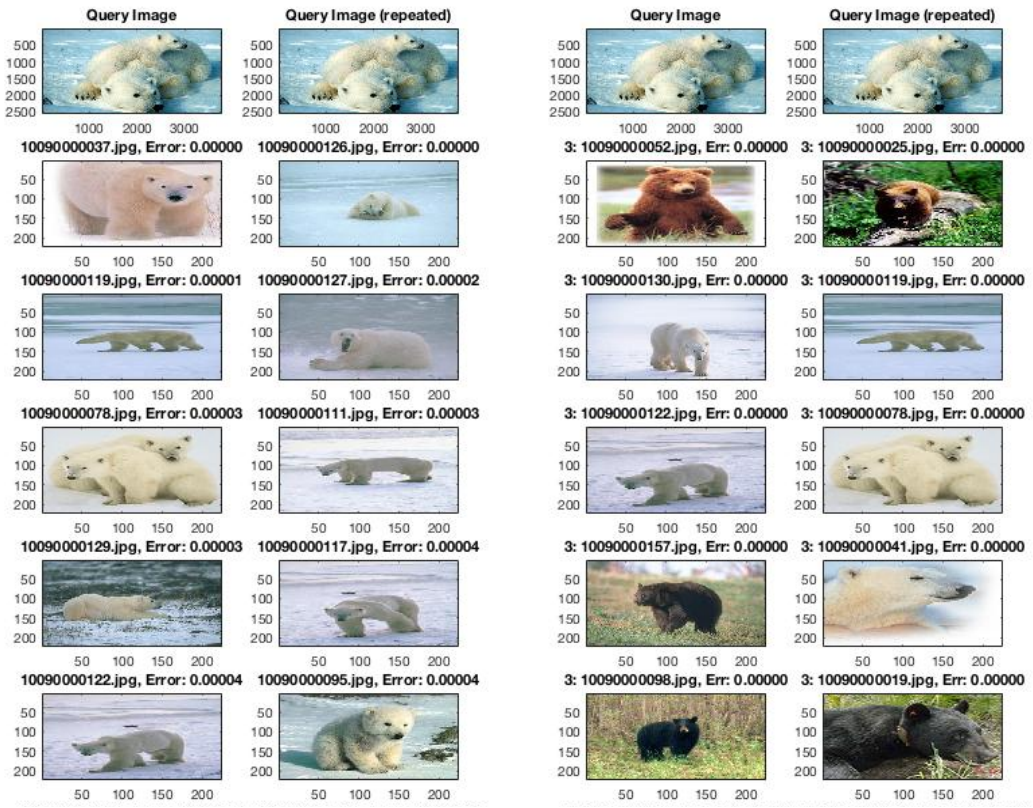

Figure 3.13: Using CBH-IBR via the original (unmodified) network (left), versus the retrained network (right)

Overall we find our results to be excellent, both qualitatively and quantitatively. Using existing networks in a novel way leverages all of the work that has gone into training DLCNNs, and with more granular image classification databases and more research into training techniques, we believe the results will improve further.

CHAPTER 4

CONCLUSION AND FUTURE WORK

As the preceding chapters show, deep neural networks excel at a variety of tasks. From correcting sequences of images from a raw video recording via a cGAN, to utilizing the output vector of a CNN in a novel manner, variations of deep neural networks prove to be powerful tools in image manipulation. In a quasi-semantic manner, these networks are able to extract important aspects of an image and perform valuable tasks that traditionally required humans to perform.

While exciting, much research remains concerning these networks and their uses. As noted above, more precision might be wrenched from the histogram method of image-based recall by training on a larger data set with more refined labeling. With this additional work in creating a more usable training set (no small task, of course) the histogram IBR method could prove to be usable in a commercial or professional context. For color correction, both classification and cGAN methods work well, but as noted above, the cGAN is likely the better choice for further research as data labeling is the big stumbling block at present. Again, with a larger, more varied data set, the cGAN color correction method could prove to be the basis for a commercially successful color correction tool.

As color correction is not a topic not much studied in Artificial Intelligence circles, not a great deal of research has been done recently on that front. Still, some work has been done recently on related topics. Zhao et al. (2018), for example, discuss real-time color correction (in camera) for CMOS-based video cameras. Sheremet et al. (2018) trained CNNs to denoise images—though

these are stills, and not video sequences. Finally, Schwartz and Giryes (2019) use a new neural net they term a DeepISP to learn the entire image processing pipeline, from raw data to final output, allowing the network to automatically cover many steps needed to make raw image data ready for human consumption. As color correction is extremely important to the film/video industry, impacting aesthetic and commercial interests, the work done here is a good first step into an area that can use much more research.

A more popular area, more research is ongoing in Image-Based Recall. Since the publication of the article on which Chapter 3 is based, much new research has focused on IBR, and in particular how to utilize information from clusters of data. For example, Sabahi, Ahmed and Swamy (2018) discuss using a perceptual image hashing function—which is in some ways analogous to our work with a histogram as it hashes images based on their content—to discover similar images. Van and Le (2018) utilize a binary cluster graph—which again functions in a similar manner to our work—to find similar content. In a somewhat more traditional method, Anjali et al. (2018) use a decision tree that outputs a classification that is then used to discover similar images.

The newer research that has been done in both IBR and image correction demonstrates continued interest in these topics. We also note that these two tasks—image-based recall and video color grading—are ripe for commercial deployment, as they both provide value to producers and consumers of image and video. The potential for the networks developed in our research to be utilized in commercial contexts is very exciting. Though much work remains to be done, especially in data collection, both methods have passed the proof-of-concept stage, and, with more data and some refinement, should prove to be viable tools for use in image-based recall and color correction.

REFERENCES

Ackley, D. H., G. E. Hinton, and T. J. Sejnowski (1985). A learning algorithm for boltzmann machines, *Cognitive science, 9(1): 147169.*

Anjali, T, et al. (2018). A Novel Based Decision Tree for Content Based Image Retrieval: An Optimal Classification Approach. International Conference on Communication and Signal Processing (ICCSP), Communication and Signal Processing (ICCSP), 2018 International Conference On, 0698.

Bar-Hillel, A., T. Hertz, N. Shental, and D. Weinshall (2003). Learning distance functions using equivalence relations,*In ICML, pages 1118.*

Bay, H., T. Tuytelaars, and L. J. V. Gool (2006). Surf: Speeded up robust features,*In ECCV (1), pages 404417.*

Bishop C.M. (1995). Neural networks for pattern recognition, *Oxford university press, Nov 23.*

Breiman, L (1996). Bagging predictors, Machine learning 24.2, pp.123-140.

Breiman, L (1998). Arcing classifier (with discussion and a rejoinder by the author),*The annals of statistics, 26(3), pp.801-849.*

CBIR (2017). *https: //github.com/aminert/CBIR*

cbires (2017). *https: //github.com/kirk86/ImageRetrieval*

Chang, H., and D.Y. Yeung (2007). Kernel-based distance metric learning for content-based image retrieval, *Image and Vision Computing, 25(5): 695703.*

Chatterjee, M., Leuski, A (2015). An Active Learning Based Approach For Effective Video Annotation And Retrieval, *arXiv preprint arXiv: 1504.07004.*

Chechik, G., V. Sharma, U. Shalit, and S. Bengio (2010). Large scale online learning of image similarity through ranking,*Journal of Machine Learning Research, 11: 11091135.*

Chen, L.C., et al (2015). Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*.

Ciresan, Dan, Ueli Meier, and Jürgen Schmidhuber (2012). Multi-column deep neural networks for image classification. Computer Vision and Pattern Recognition *CVPR*, pp. 3642-3649.

Cohn, David A., Zoubin Ghahramani, and Michael I. Jordan (1996). ”Active learning with statistical models.” Journal of artificial intelligence research.

Computerphile (2016). Deep Dream (Google). Available at: https://www.youtube.com/watch?v=BsSmBPmPeYQ.

Deng, L (2014). A tutorial survey of architectures, algorithms, and applications for deep learning, *APSIPA Transactions on Signal and Information Processing, 3: e2*.

Deng, J., et al. (2009). ImageNet: A large-scale hierarchical image database, *CVPR.*

Dietterich, T. G., Bakiri G (1995). Solving multiclass learning problems via error-correcting output codes, *Journal of artificial intelligence research 2: 263-286.*

Domeniconi, C., J. Peng, and D. Gunopulos (2002). Locally adaptive metric nearest-neighbor classification,*IEEE Trans. Pattern Anal. Mach. Intell., 24(9): 12811285.*

Evans, Claire L. Deep Dream (2016). *Frieze*, 176, p. 126.

Freund, Y., Schapire, R. E (1995). A desicion-theoretic generalization of on- line learning and an application to boosting. *European conference on computational learning theory (pp. 23-37). Springer Berlin Heidelberg, (March)*.

Gatys, L.A., A.S. Ecker, and M.Bethge (2016). Image Style Transfer Using Convolutional Neural Networks. *CVPR*.

Gibbs, J. (2018). Video Color Grading Via Deep Neural Networks. *IADIS International journal on Computer Science and Information Systems*, *13, No. 2*, 1-15.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Deep sparse rectifier networks." In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume, vol. 15, pp. 315-323.

Grother, P.J. (1995). NIST Special Database 19: Handprinted Forms and Characters Database. In National Institute of Standards and Technology.

Grove, Adam J., Schuurmans Dans (1998). Boosting in the limit: Maximizing the margin of learned ensembles,*In AAAI/IAAI, pp. 692-699.*

Guillaumin, M., J. J. Verbeek, and C. Schmid (2009). Is that you? Metric learning approaches for face identification. In ICCV, pages 498505.

Hearst, M.A., Dumais, S.T., Osman, E., Platt, J. and Scholkopf, B (1998). Support vector machines, *IEEE Intelligent Systems and their Applications, 13(4), pp.18-28.*

Hinton, Geoffrey E (1989). Deterministic Boltzmann learning performs steepest descent in weight-space. *Neural Computation* 1.1, pp. 143-150.

Hinton, G., L. Deng, D. Yu, G. E. Dahl, A.R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, *Signal Processing Magazine, IEEE, 29(6): 8297.*

Hinton, G. E., S. Osindero, and Y. W. Teh (2006). A fast learning algorithm for deep belief nets, *Neural Computation, 18(7): 15271554.*

Hinton, Geoffrey E., and Ruslan R. Salakhutdinov (2006). Reducing the dimensionality of data with neural networks. *Science* 313.5786, pp. 504-507.

Hoi, S. C. H., W. Liu, M. R. Lyu, and W.Y. Ma (2006). Learning distance metrics with contextual constraints for image retrieval,*In CVPR (2), pages 20722078.*

Iizuka, S., E. Simo-Serra, and H. Ishikawa (2016). Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (TOG)*, 35(4).

http://image-net.org (2018). [Accessed 15 Apr. 2018]

Isola, Phillip, et al (2017). Image-to-Image Translation with Conditional Adversarial Networks. *CVPR*.

Jain A. K., Vailaya, A (1996). Image retrieval using color and shape,*Pattern Recognition, 29(8): 12331244, 1996.*

Jain, P., B. Kulis, I. S. Dhillon, and K. Grauman (2008). Online metric learning and fast similarity search,*In NIPS, pages 761768.*

Ji, Shuiwang, et al (2013). 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions On Pattern Analysis and Machine Intellegence* Issue No. 01 - Jan. (Vol. 35), pp. 221-231.

Jin, R., S. Wang, and Y. Zhou (2009). Regularized distance metric learning: Theory and algorithm,*In NIPS, pages 862870.*

Jgou, H., F. Perronnin, M. Douze, J. Snchez, P. Prez, and C. Schmid (2012). Aggregating local image descriptors into compact codes,*IEEE Trans. Pattern Anal. Mach. Intell., 34(9): 17041716.*

Jones, K. S (1997). Readings in information retrieval,*Morgan Kaufmann. Chicago.*

Kearns, M. J., Vazirani, U. V (1994). An introduction to computational learning theory, *MIT press.*

Khalifa, Amine Ben, and Faezeh Tafazzoli (2013). "Content Based Image Retrieval System." https: //github.com/aminert/CBIR/blob/master/Report/FeazhAmineCBIR.pdf. *https: //github.com/aminert/CBIR.*

Krafka, Kyle J (2015). Building Real-Time Unconstrained Eye Tracking with Deep Learning. Dissertation. Suchendra Bhandarkar and W. Don Potter advisors. The University of Georgia.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks,*In NIPS, pages 11061114.*

Kulkarni, P., and Stranieri, A. (2018). Comparison of Pixel N-Grams with Histogram, Haralick's features and Bag-of-Visual-Words for Texture Image Classification. *3rd International Conference for Convergence in Technology (I2CT), Convergence in Technology (I2CT), 2018 3rd International Conference*.

Kundert-Gibbs, John (2017). Image Based Content Retrieval via Class-Based Histogram Comparisons. *Lecture Notes in Electrical Engineering, Vol. 449: IT Convergence and Security 2017, Vol. 1*, Ed. Kim, J. Kuinam, Hyuncheol Kim, Nakhoon Baek. Singapore, Springer Nature, pp. 3-10.

Kundert-Gibbs, John (2018). "Aesthetic Grading: Color Correction via Neural Networks." Kundert-Gibbs, John. *Proceedings of the Theory and Practice in Modern Computing 2018 International Conference*, Madrid, Spain.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition,*Proceedings of the IEEE, 86(11): 22782324.*

LeCun, Y., et al. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *NEURAL COMPUTATION*, *1*(4), 541–551.

Lee, J. E., R. Jin, and A. K. Jain (2008). Rank-based distance metric learning: An application to image retrieval,*In CVPR.*

Liftgammagain (2015). http://www.liftgammagain.com/forum/index.php?threads/pricing-indyfeature. 4324.

Lowe, D.G. (1999). Object recognition from local scale-invariant features,*In ICCV, pages 11501157, 1999.*

Manjunath, B. S., Ma, W.Y (1996). Texture features for browsing and retrieval of image data,*IEEE Trans. Pattern Anal. Mach. Intell., 18(8): 837842.*

MatConvNet (2017). Available at: http://www.vlfeat.org/matconvnet [Accessed 3 Apr. 2017].

McClean, Sally, Bryan Scotney and Mary Shapcott (2000). Using background knowledge in the aggregation of imprecise evidence in databases. *Data & Knowledge Engineering* Volume 32, Issue 2, pp. 131-143.

Mian, A. S., Y. Hu, R. Hartley, and R. A. Owens (2013). Image set based face recognition using self-regularized non-negative coding and adaptive distance metric learning,*IEEE Transactions on Image Processing, 22(12): 52525262.*

Mikolov, et al (2013). Efficient Estimation of Word Representations in Vector Space. *NIPS.*

Mirza, M., and S. Osindero (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Mitchell, T. M (1997). Machine Learning, *McGraw-Hill 1st edition (October 1).*

Mitro, Joani (2016). Content-based image retrieval tutorial, *arXiv preprint arXiv: 1608.03811. https: //github.com/kirk86/ImageRetrieval.*

Moreira, M., Mayoraz E (1998). Improved pairwise coupling classification with correcting classifiers,*In European conference on machine learning, pp. 160-171,Springer Berlin Heidelberg.*

Nigrin, Albert (1993). Neural networks for pattern recognition,*MIT Press.*

Norouzi, M., D. J. Fleet, and R. Salakhutdinov (2012). Hamming distance metric learning,*In NIPS, pages 10701078.*

Oliva, A.,Torralba, A (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope,*International Journal of Computer Vision, 42(3): 145175.*

Oliva, A.,Torralba, A (2002). Scene-centered description from spatial envelope properties,*In Biologically Motivated Computer Vision, pages 263272.*

Olshausen, Bruno A., and David J. Field (1997). Sparse coding with an overcomplete basis set: A strategy employed by VI? *Vision Research* 37.23, pp. 3311-3326.

Ortega-Binderberger, M. and Mehrotra, S (2003). Relevance feedback in multi- media databases,*Handbook of video databases design and applications, pp.1-28.*

Pessa, E. (Ed.). (1988). *Neural networks and natural intelligence*.

Philbin, J. et al. (2007). Object retrieval with large vocabularies and fast spatial matching,*In CVPR.*

Pix2Pix (2017). Available at: https://github.com/phillipi/pix2pix [Accessed 12 Dec. 2017].

Porkaew, Kriengkrai,and Kaushik Chakrabarti (1999). Query refinement for multimedia similarity retrieval in MARS, *Proceedings of the seventh ACM international conference on Multimedia (Part 1), pp. 235-238. ACM.*

Quinlan, J. Ross Induction of decision trees, *Machine learning 1.1 (1986): 81-106.*

Quinlan, J. R. Bagging, boosting, and C4. 5. *In AAAI/IAAI, Vol. 1 (1996), pp. 725-730.*

Radford, A., L. Metz, and S. Chintala (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rahman, M., et al (2017). Skin lesions classification based on color plane-histogram-image quality analysis features extracted from digital images. *IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Circuits and Systems (MWSCAS), 2017 IEEE 60th International Midwest Symposium*.

Reinhard, E., et al (2001). Color transfer between images. *IEEE Computer Graphics and Applications*, 21:34–41.

Rosenblatt, Frank (1957). The Perceptron, A Perceiving and Recognizing Automaton. Project Para Report No. 85-460-1, Cornell Aeronautical Laboratory (CAL).

Rosenblatt, Frank (1960). Perceptron simulation experiments, *Proceedings of the IRE 18*, 3 March, 301-309.

Rosenblatt, Frank (1966). Comparison of a five-layer perceptron with human visual performance, in *Natural Automata and Useful Simulations*, pp. 139, Spartan Books.

Sabahi, F., Ahmad, M. O., Swamy, M. N. S (2018). Content-based Image Retrieval using Perceptual Image Hashing and Hopfield Neural Network. (2018). 2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS), Circuits and Systems (MWSCAS), 2018 IEEE 61st International Midwest Symposium, 352.

Salakhutdinov, R., and G. E. Hinton (2009a). Semantic Hashing, *Int. J. Approx. Reasoning, 50(7): 969978.*

Salakhutdinov R., and G. E. Hinton (2009b). Deep boltzmann machines, *In AISTATS, pages 448455.*

Salakhutdinov, R., A. Mnih, and G. E. Hinton (2007). Restricted boltzmann machines for collaborative filtering, *In ICML, pages 791798.*

Schwartz, E., Giryes, R., Bronstein, A.M (2019). DeepISP: Toward Learning an End-to-End Image Processing Pipeline. IEEE Transactions on Image Processing, Image Processing, IEEE Transactions on, IEEE Trans. on Image Process, (2), 912.

Sheremet, et al. (2018). Convolutional Neural Networks for Image Denoising in Infocommunication Systems. 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Problems of Infocommunications. Science and Technology (PIC S&T), 2018 International Scientific-Practical Conference, 429.

Sivic, J., et al. (2005). Discovering objects and their localization in images,*In ICCV, pages 370377.*

Swain, M. J., & Ballard, D. H. (1992). Indexing via color histograms. In *Active Perception and Robot Vision* (pp. 261-273). Springer Berlin Heidelberg.

Tano, S., Oyama, T., & Arnould, T. (1982). Deep combination of fuzzy inference and neural network in fuzzy inference software - FINEST. *FUZZY SETS AND SYSTEMS*, (2), 151.

Tong, Simon, and Edward Chang (2001). Support vector machine active learning for image retrieval, *Proceedings of the ninth ACM international conference on Multimedia (MM'01), pgs.107-118 ACM.*

Torch (2017). Available at: http://torch.ch [Accessed 3 Dec. 2017].

Turing, A., and Copeland, B. J. (2004). *The Essential Turing*. Oxford: Clarendon Press.

Van, T.T. & Le, T.M. (2018). Content-based image retrieval based on binary signatures cluster graph. Expert Systems, 35(1), 1.

Vedaldi, Andrea and Andrew Zisserman (2017). Oxford Geometry Group VGG Convolutional Neural Networks Practical. Available at: http://www.robots.ox.ac.uk/~vgg/practicals/cnn/index.html#part1-4.

Vedaldi, A (2013). MatConvNet. Convolutional Neural Networks for MAT-LAB. *Proceedings of the ACM Intl. Conference on Multimedia (MM'13)*. http: //www.vlfeat.org/matconvnet.

Vedaldi, Andrea, Karel Lenc and Joao Henriques (2016). Oxford Geometry Group VGG CNN Practical: Image Regression Practical. Available at: http://www.robots.ox.ac.uk/~vgg/practicals/cnn-reg/index.html#part-35-learning-a-larger-model-using-the-gpu.

Vinyals, Toshev, Bengio, and Erhan (2015). Show and Tell: A Neural Image Caption Generator. *CVPR*.

http://www.vision.caltech.edu/Image_Datasets/Caltech256 (2018) [Accessed 5 November 2018]

Walker, J. L., & Hill, E. v. K. (1993). *Back propagation neural networks for predicting ultimate strengths of unidirectional graphite/epoxy tensile specimens [microform] / James L. Walker, Eric v. K. Hill*. [Washington, DC : Springfield, Va. : National Aeronautics and Space Administration ; National Technical Information Service, distributor]

Wang, D., S. C. H. Hoi, P. Wu, J. Zhu, Y. He, and C. Miao (2013a). Learning to name faces: a multimodal learning scheme for search-based face annotation, *In SIGIR, pages 443452.*

Wang, Z., Y. Hu, and L.-T. Chia (2013b). Learning image-to-class distance metric for image classification,*ACM TIST, 4(2): 34.*

Weinberger, K. Q., J. Blitzer, and L. K. Saul (2005). Distance metric learning for large margin nearest neighbor classification,*In NIPS.*

Werbos, P.J. (1992). Neural networks and the human mind: new mathematics fits humanistic insight. *1992 IEEE International Conference on Systems, Man, and Cybernetics, Systems, Man and Cybernetics*.

Wichmann, F. A., Sharpe, L. T., & Gegenfurtner, K. R. (2002). The contributions of color to recognition memory for natural scenes. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28(3), 509.

Wood, Kristin L., and Erik K. Antonsson (1989). Computations with Imprecise Parameters in Engineering Design: Background and Theory. *ASME Journal of Mechanisms, Transmissions, and Automation in Design* Volume 111, Number 4, pp. 616-625.

Wu, J. and J. M. Rehg (2011). Centrist A visual descriptor for scene categorization, *IEEE Trans. Pattern Anal. Mach. Intell, 33(8): 14891501.*

Wu, Leejay, et al (2000). Falcon Feedback adaptive loop for content-based retrieval. *No. CMU-CS-00-142. CARNEGIE-MELLON UNIV PITTS- BURGH PA SCHOOL OF COMPUTER SCIENCE.*

Wu, L., S. C. H. Hoi, and N. Yu (2010). Semantics-preserving bag-of- words models and applications,*IEEE Transactions on Image Processing, 19(7): 19081920.*

Wu, L. and S. C. H. Hoi (2011). Enhancing bag-of-words models with semantics-preserving metric learning,*IEEE MultiMedia, 18(1): 2437.*

Yang, J., Y.-G. Jiang, A. G. Hauptmann, and C.W. Ngo (2007). Evaluating bag- of-visual-words representations in scene classification,*In Multimedia Information Retrieval, pages 197206.*

Yoon, H., Nang, J.H., Maeng, S.R. (1990). A distributed backpropagation algorithm of neural networks on distributed-memory multiprocessors. *The Third Symposium on the Frontiers of Massively Parallel Computation, Frontiers of Massively Parallel Computation*.

Yoshua Bengio and Yann LeCun (2007). Scaling learning algorithms towards AI. Bottou, L. and Chapelle, O. and DeCoste, D. and Weston, J. (Eds), *Large-Scale Kernel Machines*, MIT Press.

Zhang, L., Y. Zhang, X. Gu, J. Tang, and Q. Tian (2014). Scalable similarity search with topology preserving hashing,*IEEE Transactions on Image Processing, 23(7): 30253039.*

Zhang, Richard, Phillip Isola and Alexei A. Efros (2016). Colorful Image Colorization. Available at: http://richzhang.github.io/ colorization/resources/colorful_eccv2016.pdf.

Zhang, Y., L. Zhang, and Q. Tian (2014). A prior-free weighting scheme for bi- nary code ranking. IEEE Transactions on Multimedia, 16(4): 11271139, 2014.

Zhao, Zhichao, et al (2018). Research on Color Correction Based on CMOS Image Sensor. International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), Intelligent Transportation, Big Data & Smart City (ICITBS), 2018 International Conference on, ICITBS, 699.

Zhou, X.S. and Huang, T.S (2001). Comparing discriminating transformations and SVM for learning during multimedia retrieval*In Proceedings of the ninth ACM international conference on Multimedia (pp. 137-146). ACM, October.*

Zhou, X.S. and Huang, T.S (2003). Relevance feedback in image retrieval: A comprehensive review,*Multimedia systems, 8(6), pp.536-544,2003.*

Zisserman, Arxiv (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ILSVRC.*

APPENDIX A

THE LIST OF 50 QUERY IMAGES FOR CHAPTER 2

Table 2 lists descriptions of the 50 query images in the experiment described in this paper. All query images are completely independent of the image database used to retrieve images. All images are similar to one or more image categories in the image database, with at least 80 images in the database a close match.

Table A.1: Images and descriptors of images used to test the four IBR search engines.

Image Number	Image Type
1	barn (distant)
2	bear (brown)
3	smokestack (1)
4	bus (mega)
5	butterfly (monarch)
6	castle (medieval)
7	cat (orange tabby)
8	Fire Truck
9	flowers (closeup)
10	horse (brown)
11	military jet (1)
12	lake + mountains
13	passenger jet

14	snake
15	sailboat (plus land)
16	soccer (professional)
17	smokestack (multiple)
18	tiger (standing)
19	waterfall (small)
20	mansion (white house)
21	wine glass (with wine)
22	barn (medium)
23	sheet music
24	palm tree
25	bus (city transit)
26	butterfly (unknown)
27	castle (neuschwanstein)
28	cat (orange tabby)
29	classic car (blue)
30	cricket (sport)
31	panther
32	military jet (3)
33	flowers (closeup)
34	mansion
35	mountains (sky, clouds)
36	passenger jet

37	bear (polar)
38	gorilla
39	sailboat (plus land)
40	soccer (amateur)
41	tiger (recumbent)
42	waterfall (medium)
43	wine glasses (2)
44	beer glass (with beer)
45	horse (brown)
46	lake + hills
47	coffee mug
48	dogs (3)
49	stream
50	mandolin