IADIS International Journal on Computer Science and Information Systems
Vol. 13, No. 2, pp. 1-15
ISSN: 1646-3692

VIDEO COLOR GRADING VIA DEEP NEURAL NETWORKS

John L. Gibbs
The University of Georgia, USA

ABSTRACT

The task of color grading (or color correction) for film and video is significant and complex, involving aesthetic and technical decisions that require a trained operator and a good deal of time. In order to determine whether deep neural networks are capable of learning this complex aesthetic task, we compare two network frameworks—a classification network, and a conditional generative adversarial network, or cGAN—examining the quality and consistency of their output as potential automated solutions to color correction. Results are very good for both networks, though each exhibits problem areas. The classification network has issues with generalizing due to the need to collect and especially to label all data being used to train it. The cGAN on the other hand can use unlabeled data, which is much easier to collect. While the classification network does not directly affect images, only identifying image problems, the cGAN, creates a new image, introducing potential image degradation in the process; thus multiple adjustments to the network need to be made to create high quality output. We find that the data labeling issue for the classification network is a less tractable problem than the image correction and continuity issues discovered with the cGAN method, which have direct solutions. Thus we conclude the cGAN is the more promising network with which to automate color correction and grading.

KEYWORDS

Color Correction, Generative Adversarial Neural Network

1. INTRODUCTION

Color grading, which is 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 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.[1] The art of color grading and correction is to make every shot look "good" (an 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 producers, 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, both 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. 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

[1] 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.

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 Imagenet 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 search 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 previous work we explored the semantic issue of image-based recall (IBR) (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 based on input images, using 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.

3. ASSET COLLECTION

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 Kundert-Gibbs 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 an opinion by 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 Kundert-Gibbs 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, Kundert-Gibbs 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 had no discernable effect on the network’s success. For the cGAN, input aspect ratio is not important, so we utilized the properly sized 456X256 pixel images to generate image pairs that are 912X256 pixels.

4. EXPERIMENTAL SETUP

We simultaneously explored 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 would be able to fool a discriminator network as it compared the generator's output with the target (corrected) image. While both options use deep CNNs, the two paths are fundamentally different in their approach.

4.1 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 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 which 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 for our focus on low frequency issues within the test images.

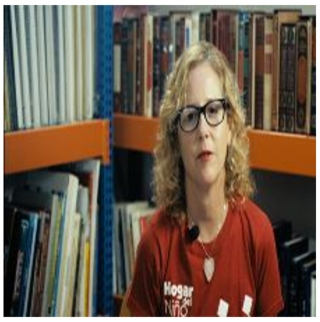

Figure 1. A sample input image with horizontal dimensions compressed to create a 224X224 square for the classification network

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 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 move the orange 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 (see Results, below).

4.2 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, shown in Figure 2. These image pairs were then fed to the cGAN for training.

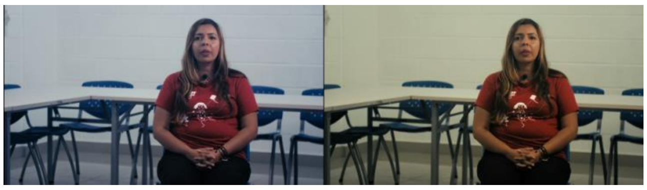

Figure 2. An example image pair (uncorrected, very blue, image on the left; corrected, or target image on the right) that is fed into the conditional generative adversarial network

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 are a vast 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 following images might be adjusted differently from each other, thus causing images to flicker as they are presented at 24 or more frames per second.

For the cGAN, another significant concern was 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 looked slightly softer 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. 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)[2] 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.

4.3 General

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 does well in these areas. 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 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).

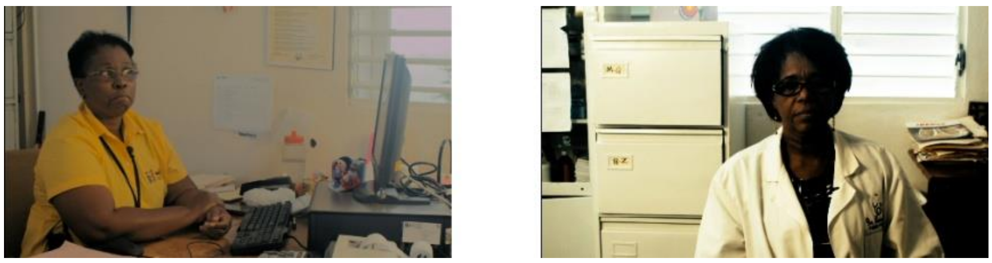

Figure 3. Two source images showing the low frequency nature of color correction issues. The left image has its white point set 100% too low, while the right image has its contrast set 100% too high

[2] 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.

5. 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 4.1 discusses results for the classification network while 4.2 discusses results for the conditional generative adversarial network.

5.1 Classification Network

For our classification network, optimal results for the training data was found very quickly, within 30 epochs of retraining the modified VGG-f network. As shown in Figure 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 5. For the trained network, nearly all errors made were in neighboring classifications. 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. 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 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 1. Error rates on validation set, including correct and "nearest neighbor" errors in classification of image problems

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

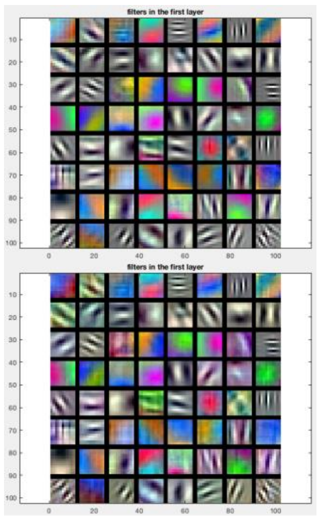

Figure 4. First layer convolution filters before retraining (top) and after 30 epochs of retraining (bottom)

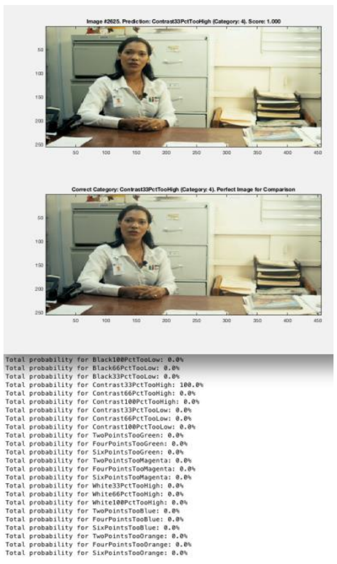
Image #2625. Prediction: Contrast33PctTooHigh (Category: 4). Score: 1.000
Correct Category: Contrast33PctTooHigh (Category: 4). Perfect Image for Comparison

```
Total probability for Black100PctTooLow: 0.0%
Total probability for Black66PctTooLow: 0.0%
Total probability for Black33PctTooLow: 0.0%
Total probability for Contrast33PctTooHigh: 100.0%
Total probability for Contrast66PctTooHigh: 0.0%
Total probability for Contrast100PctTooHigh: 0.0%
Total probability for Contrast33PctTooLow: 0.0%
Total probability for Contrast66PctTooLow: 0.0%
Total probability for Contrast100PctTooLow: 0.0%
Total probability for TwoPointsTooGreen: 0.0%
Total probability for FourPointsTooGreen: 0.0%
Total probability for SixPointsTooGreen: 0.0%
Total probability for TwoPointsTooMagenta: 0.0%
Total probability for FourPointsTooMagenta: 0.0%
Total probability for SixPointsTooMagenta: 0.0%
Total probability for White33PctTooHigh: 0.0%
Total probability for White66PctTooHigh: 0.0%
Total probability for White100PctTooHigh: 0.0%
Total probability for TwoPointsTooBlue: 0.0%
Total probability for FourPointsTooBlue: 0.0%
Total probability for SixPointsTooBlue: 0.0%
Total probability for TwoPointsTooOrange: 0.0%
Total probability for FourPointsTooOrange: 0.0%
Total probability for SixPointsTooOrange: 0.0%
```

Figure 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 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.

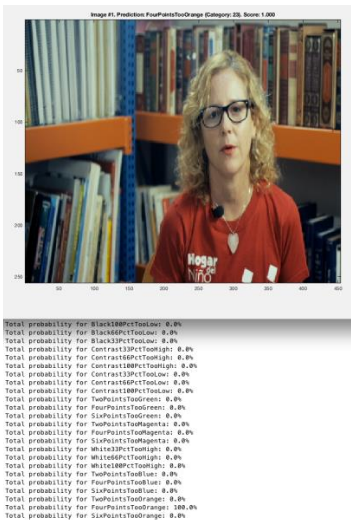
Image #1. Prediction: FourPointsTooOrange (Category: 23). Score: 1.000

```
Total probability for Black100PctTooLow: 0.0%
Total probability for Black66PctTooLow: 0.0%
Total probability for Black33PctTooLow: 0.0%
Total probability for Contrast33PctTooHigh: 0.0%
Total probability for Contrast66PctTooHigh: 0.0%
Total probability for Contrast100PctTooHigh: 0.0%
Total probability for Contrast33PctTooLow: 0.0%
Total probability for Contrast66PctTooLow: 0.0%
Total probability for Contrast100PctTooLow: 0.0%
Total probability for TwoPointsTooGreen: 0.0%
Total probability for FourPointsTooGreen: 0.0%
Total probability for SixPointsTooGreen: 0.0%
Total probability for TwoPointsTooMagenta: 0.0%
Total probability for FourPointsTooMagenta: 0.0%
Total probability for SixPointsTooMagenta: 0.0%
Total probability for White33PctTooHigh: 0.0%
Total probability for White66PctTooHigh: 0.0%
Total probability for White100PctTooHigh: 0.0%
Total probability for TwoPointsTooBlue: 0.0%
Total probability for FourPointsTooBlue: 0.0%
Total probability for SixPointsTooBlue: 0.0%
Total probability for TwoPointsTooOrange: 0.0%
Total probability for FourPointsTooOrange: 100.0%
Total probability for SixPointsTooOrange: 0.0%
```

Figure 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

5.2 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 7, the sample output image is nearly indistinguishable from the target (ground truth) image. The right-most image shows the results of subtracting the two images in Photoshop (via the difference layer mode). That this image is nearly completely black, 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.

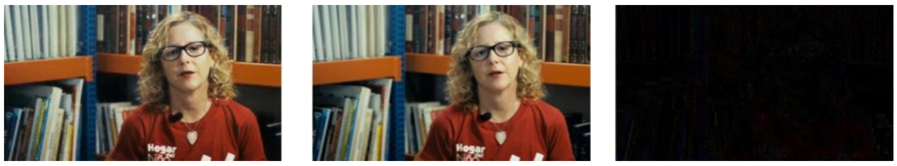

Figure 7. cGAN output, left, compared to the target image, center. Right is the difference between the two images (via difference layer mode in Photoshop). The nearly black result shows that most pixels have nearly the same value (the image on left is an unusually poor output, thus showing at least some 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. 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 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.

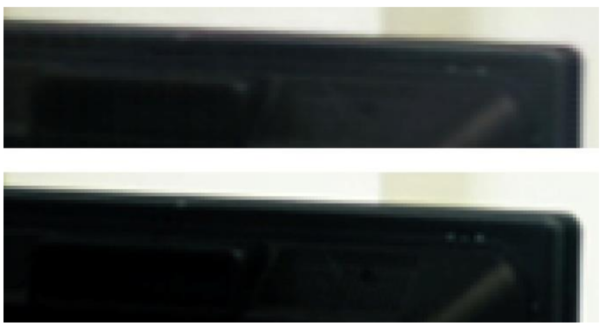

Figure 8. The cGAN run on a standard sized image (100% in X and Y), on top, versus a 2X oversampled image (200% in X and Y), on bottom. A blown up portion of the image is shown here in order to reveal fine detail

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 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 and the individual image softness is not apparent.

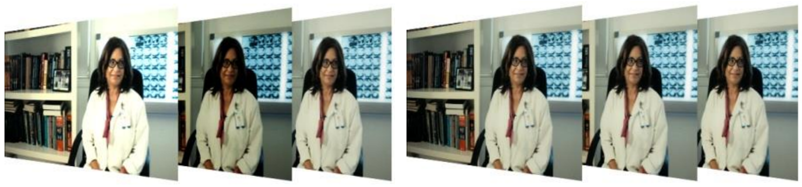

Figure 9. Multi-frame differences produce a flickering effect, left (greatly exaggerated for clarity), while frame blending, right, smooths out frame-to-frame differences to produce a more pleasing sequence of images

Our other solution to the variability of concurrent 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. Figure 10 shows a portion of an image sequence (with random flipping), indicating that this solution creates consistent output images over a sequence.

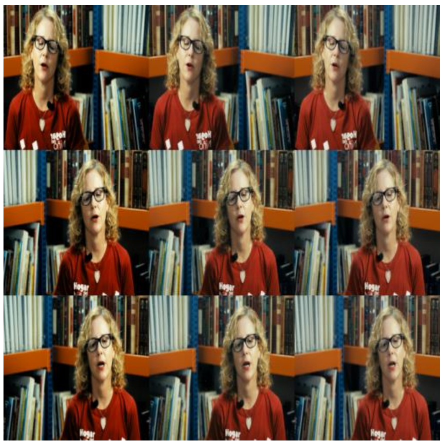

Figure 10. Training the cGAN on image sequences as a group produces a more consistent look. Left is the uncorrected input, middle the cGAN output, and right, the corrected target image. Note the random image flipping in the sequence

Very importantly, the cGAN generalizes well. Given image types the network has not trained on at all, as in Figure 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.

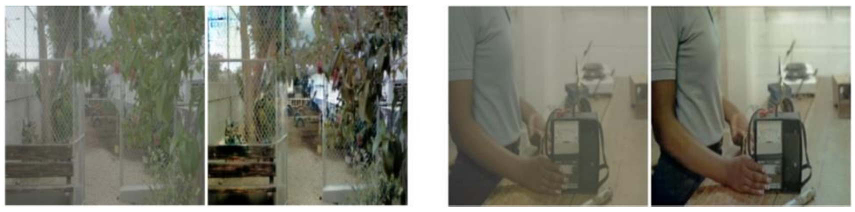

Figure 11. Given totally new types of input images, the cGAN produces high quality results. Original uncorrected images are on the left, while corrected output images are on the right

6. DISCUSSION AND CONCLUSION

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 $20 \bullet 100$, 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 long 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 images. 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.

REFERENCES

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.

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

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

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

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.

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

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

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).

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

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.

Krizhevsky, Alex, Ilya Sutskever and Geoffrey E. Hinton (2012). ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*.

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.

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

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.

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*.

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.

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

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

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

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

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, 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*.

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

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.

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

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