Image Colorization

By

Eli Harris, Tristan Fullmer, Ethan Nelson

Introduction

A quick google search reveals that colored photography began in the 1860s. The quality and use of taking colored images continued to progress over time. Likewise, the first colored film came out in the early 20th century. However black and white films continued to be produced even after the 1950s.

Many black and white photos would give one additional insight by providing the colored complement. One example is old family history photos. Although the black and white image gives some information, it is missing important details such as hair and eye color. We use a neural network machine learning algorithm to ‘colorize’ the black and white images.

We used images from several repositories including the following:
https://www.floydhub.com/emilwallner/datasets/colornet/2/images,

https://unsplash.com/collections

The floydhub repository has both black and white and colored images for the training and test sets. We augmented these sets with additional images from the unsplash repository. From the unsplash repository, we obtain images from a variety of themes to diversify the types of images we train on. Some of the themes include:

- Winter: Mostly white images with some color
- Happiness: bright colorful images with lots of colors
- Moody: dark black/greyish images with a little color
- OneColor: images that have mostly one color
- Neon: images that are filled with bright neon colors

By diversifying the training set, we expected to obtain better results.

Data Preparation

In order to pre-process our data, we need to get the RGB values out of the image then convert the RGB values to LAB. This is important because the LAB separates the color from the black and white. By removing the color we are able to create a testing and training sets.

The algorithm we are using was designed to take photos that are 256 by 256 pixels. Rather than generalize the code to fit any sized picture, we decided to scale and crop the photos. We used InfranView64 to scale and crop the photos to the correct size for the algorithm. There are several reasons to scale and crop the photos rather than generalizing the algorithm to take in any size photos. The foremost reason is that bigger photos add a significantly larger computational cost than the scaled ones. Additionally, the scaled photos do not significantly reduce the information of the data.

Our dataset is non-trivial in a few ways, one being the fact that the values are stored in a 3D array. Neural networks, at least to our understanding, are not easily able to handle 3D arrays. One way we are going to try to overcome this is to convert our data to a 2D array or a vector then revert the results back to a 3D array

Mining/learning from the data

In order to learn patterns from our data, we had to find images with the same number of data points, meaning that they were the same size. While researching image processing we found out that Convolutional Neural Networks (CNN) seemed most common and efficient. The reason behind this is because a CNN can take in a three-dimensional object, find patterns in each layer and use that to predict a single layer. In short, we were trying to take a one-dimensional array and use that layer to predict a second and third layer. The way that neural networks work is that they use weights, meaning we can have a weight for every layer or input or output, etc, and in short it is easier to train a series of weights on an array rather than a singular algorithm on every single data point in the array. The parameters that we used included how many times that we trained the weights (epochs), the size of the image and the activation functions.

The first thing that we have tried is converting our data into a 2d array, however, it has not been working too well. In the end, we did get the images to convert to arrays and train the network on that. We tried a few different approaches to how we set up our CNN. they way that CNNs work makes them very particular in how you add the layers and how your model is put together. Training a CNN also is very time consuming and computationally expensive. We tried a few approaches into saving our weights, one saving them in binary and another using h5, which is python’s standard library.

The biggest challenge that we faced was getting the neural network to correctly predict a colorized image. It was difficult to get the network to predict itself, we grayscaled an image and used that as training and testing data. After that, because there are such a variety of colors,

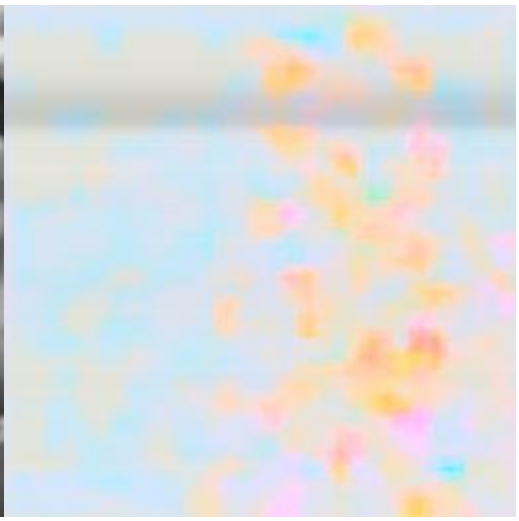
shades, and brightnesses that it was immensely difficult to train a CNN on a dataset of that size. In our training set we used about 550 images and tested them at 10,000 epochs, or cycles it takes eight to nine hours on a GPU to run each test.

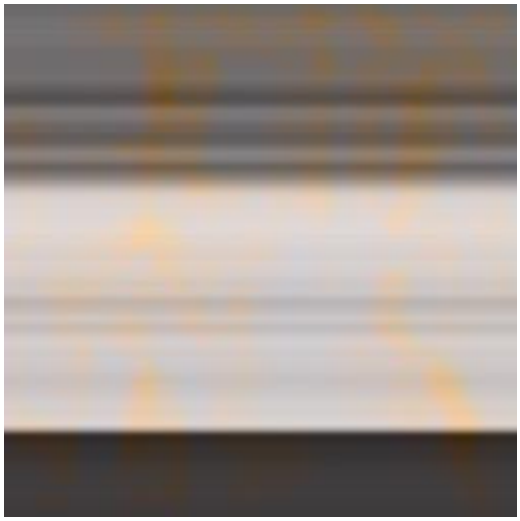
Results

We successfully implemented a neural network that imports and exports images. However, as shown below, the resulting algorithm predictions are underwhelming. The CNN was able to add some color here or there but the CNN colored images are hardly recognizable with their black and white input image.

This was one of our first test images, notice the water and the man's silhouette.

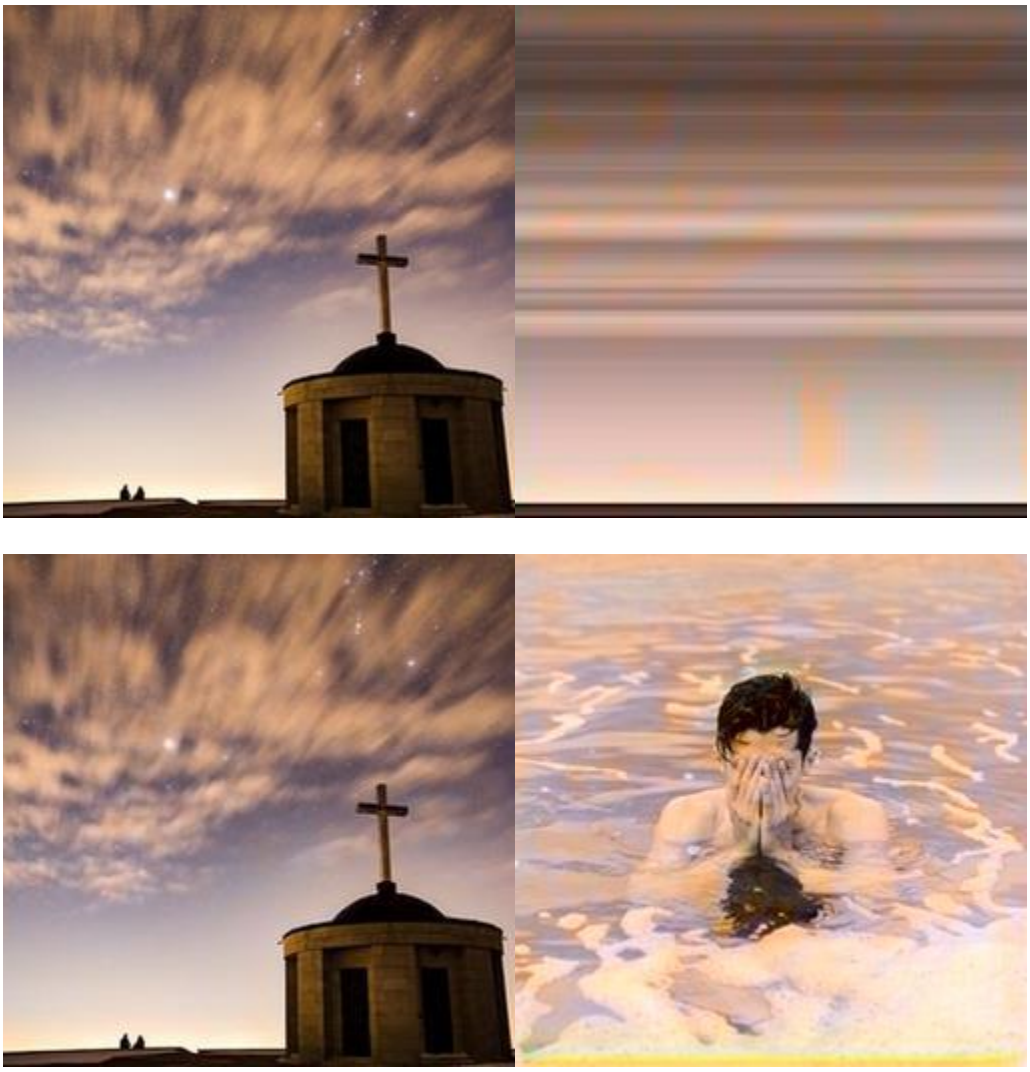
Here notice the child crown in the second image and the outline in the third





This image gave us a good insight into color schemes. You can see the temples outline very faintly in the second image. The third image is us training the network on one image with 500 epochs and testing on the grayscale version to see if we get the same image back. This was

letting us know that our network was working. The fourth image was training on the temple image but predicting the swimmer image shown above. We did this to make sure that our network could use its gathered colors to predict other images.



Conclusions

Our results could have a number of applications. The first and most obvious would be for historical organizations to be able to view old black and white pictures in color. If this concept were taken and applied to video, we could achieve two things, it could be used on nighttime security cameras to be able to see in full color even at night time as well as using this to colorize black and white movies. The last application is training on an image with a certain color schema and using that as a filter to create other images with the same color scheme. As a result of our

work, we feel that business owners in our surrounding area would have something more to offer the community. There are many businesses whose main purpose is family history. It would be a great opportunity for these businesses to offer a colorized version of a black and white photo taken of a deceased great grandparent or perhaps an ancestor.

Our results are very interesting. As described earlier, our network can only perform as well as the images trained on, and as there are a great variety of colors and shades we do not have the image dataset, or the time (as data could be gathered) to train a network being capable of correctly predicting a random image 100%. Some of our images ended up with random colors, as we did not have the correct shade trained, and some ended up as blobs of color are just being the silhouette of the image.

One ethical issue is discussing how far you can take the validity of computer coloring. If a security camera is used as evidence of a crime, and the color of something is important, it raises the question of the court's level of trust in the computer's prediction.

Lessons Learned

One thing that we learned from this project was the complexity of Convolutional Neural Networks and the spaces that it uses. There are a lot of concepts that go into CNNs and they are difficult to compose. You can create a CNN to find vertical or horizontal lines. You can use them to classify text. It was interesting to learn how rearranging the layers can change the whole model, even if you are using the same images.

If we had the opportunity to redo this project we would try to make our algorithm training more efficient. We are using Keras and tensorflow to run our network and it would have been nice to have additional time in research to find a more efficient way to run our algorithm. For example, we are iterating through a list and training on the arrays inside of it, but if we put our images together into a massive array then we might be able to train on all of the data all at once rather than individually.