Dylan Webb
CS 474 Final Project

Introduction

Neural networks have emerged as the quintessential algorithm for tackling complex visual tasks, and because the resulting images can be easily appreciated by the public, their success has been widely discussed. The deep learning projects in CS 474 that captured my interest the most this semester were those dealing with image generation and manipulation, so I decided to design a photo colorizer for my self-governed, final project: a deep neural network that realistically adds color to black and white photos.

Dataset and Problem Discussion

I approached this problem as a supervised regression problem. Black and white photos were fed through a network that predicted an output value for each pixel, thereby coloring them. The resulting image was then compared with the actual color version and the loss was computed between each pixel. This is a well-researched DNN task that can be performed remarkably well (http://www.whatimade.today/our-frst-reddit-bot-coloring-b-2/), and after reading Emil Wallner's guide to colorizing photos (https://emilwallner.medium.com/colorize-b-w-photos-with-a-100-line-neural-network-53d9b4449f8d), I was convinced that the task could be accomplished using a U-Net architecture.

To train my U-Net, I used 1000 photos of children at play from Kaggle (https://www.kaggle.com/datasets/sapal6/motion) due to their colorful and varied subject matter (kids, toys, outdoor, indoors, etc.). Even more importantly, most of the photos in this dataset are of humans; the most common application of this algorithm is to color old photos that were only taken in black and white, especially old photos of family members. I hoped that by choosing a dataset with such a colorful palette and diverse set of people, locations, and objects, the model could learn how to color many different scenarios, even though these photos do not accurately represent the type of photos taken back in the black and white era.

Model Design and Performance

My colorizer model developed over multiple iterations. I used the Adam optimizer and MSE loss function—the same setup as Emil Wallner—and began by training a U-Net commonly used for images (https://arxiv.org/pdf/1505.04597.pdf) on all 1000 photos of children at play after converting them to black and white (there was no need for testing data since I tested by visual inspection). Unfortunately, the predicted colors were very washed-out (see Figure 1a). Since the model was unable to learn how to add all the color at once, I decided to break the problem into smaller steps. I did so by interpolating between the black and white and colored version of each photo to obtain nine additional intermediate photos where color was progressively introduced. After training the model on these smaller steppingstone photos, I ran a black and white photo through the model ten times to obtain its final, colored version. This time, the model's predictions were colorful but monochromatic (see Figure 1b); the model was simply layering the same amount of color at each step, essentially saturating the initial, washed-out prediction.

To remedy this, I decided to create ten models, each trained on a different step in the interpolation. For example, the first model handled the step from black and white to one-tenth color and the last model handled the step from nine-tenths to full color. The result finally had varied amounts of color applied to each part of the photo, but one color was still dominating (see Figure 1c). For the final form of the model, I split each photo into its three component parts—red, green, and blue—and performed the same process as before, with ten neural nets for each color: in total, 30 separate U-Nets and 930 million parameters. Combined, these models reduced the MSE loss to .002 over eight epochs and finally produced photos with a balanced application of blues, greens, and reds in separate locations (see Figure 1d). I also tested it on a black and white photo taken in the 1940s of my great-great-uncle, Frank Bradford, with stunning results (see Figure1e).

Conclusion

Coloring photos was a harder problem than I initially thought, but deep neural networks are especially well-suited to solve this kind of problem. Although my model is certainly weaker than state-of-the-art methods, it took only 1000 photos and under one billion weights to produce reasonably colored photos. With more training data and deeper U-Nets, I think the approach of splitting the prediction task among multiple independent models could produce strikingly accurate results.

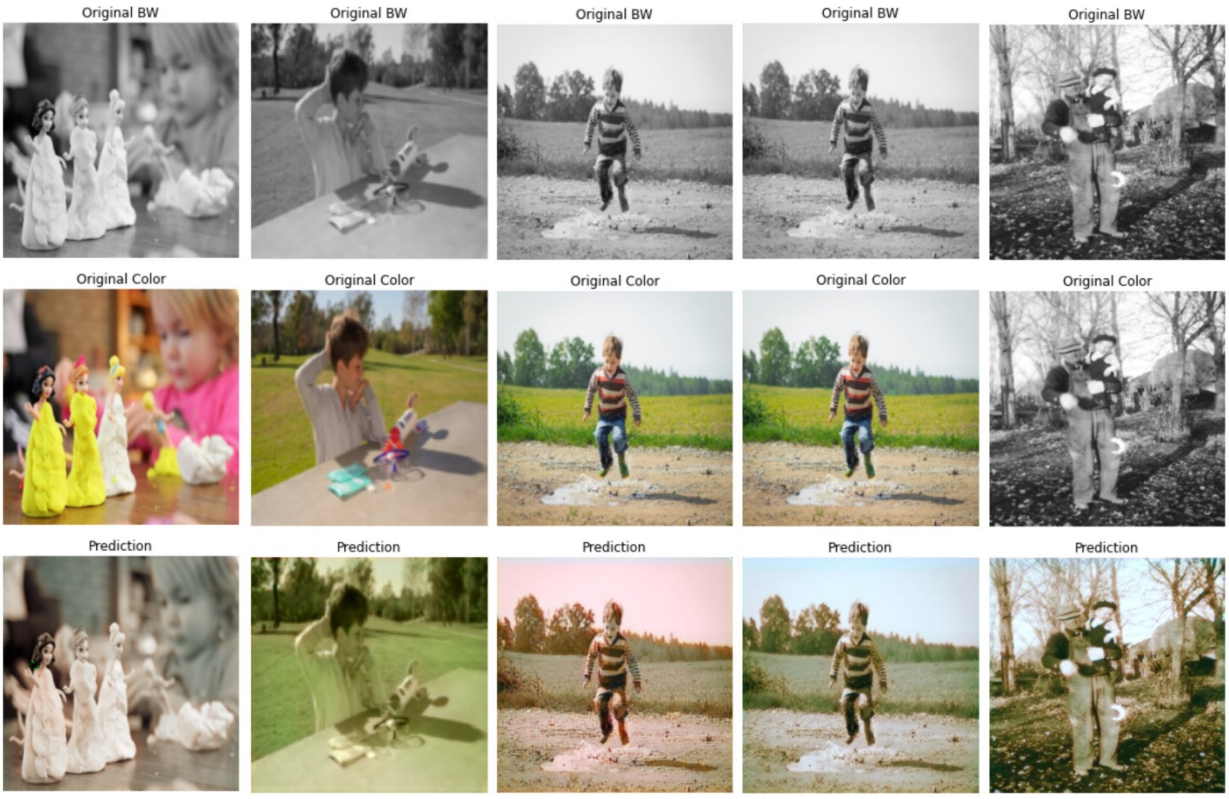

Figure 1. Comparing photos in black and white, color, and with predicted coloring.

Dylan Webb
CS 474 Final Project

Time Log – Total: 33 hours

Date	Duration	Description
October 29	2 hours	Downloaded images and prepared code for displaying them in color and black and white
December 5	2 hours	Set up github repository and automatic photo download/unzip into Colab
December 6	1 hour	Read Emil Wallner and Amir Avni guides
December 7	4 hours*	Prepared initial model and ran for the first time, obtaining muted colors
December 8	5 hours*	Worked on diffusion-type approach, obtaining monochromatic photos
December 9	6 hours*	Reworked code to train multiple models on interpolations of photos, saving weights to Google Drive for quicker testing
December 10	7 hours*	Split up images, modified models to train on a single color, and tracked losses for plotting
December 11	3 hours*	Continued tweaking model and optimizing, comparing to other possible setups
December 16	3 hours	Composed final write-up for submission

*Includes time spent debugging