Colorization

Anirban Chakraborty

Representing the Process:

I represented the coloring process as “the parts equal the whole”. I create a dataset of RGB values extracted from the images, and then I pad the dataset with RGB values with a small width of 3 to create an inflated dataset of RGB values on my own. This data consists of bright and saturated color values that I purposefully kept in my when I picked my small sample of images in order to make the model easily recognize the difference in the shade of grey and what the environment is. I wanted to colorize environments due to homogeneity in standard colors (e.g blue sky, green grass, white clouds) for a better fit and predict. As grey values usually do not have enough information to reconstruct the RGB values of the same cell, I made sure my model tried to match similar data from grey to RGB by doing a simple fit for all 3 color arrays, so that my updated training data can be used with my model to predict if the new color arrays matches with the training data, and update the BW array of the created image with the new color arrays.

Data:

I am getting my dataset from padding RGB values from the extracted respective color values from my small sample of images. This method may be convenient since I can’t use

standard datasets such as MS-COCO, but artificially padding the values may have corrupted my color arrays of the extracted images. I found this aspect difficult because I didn't want to amass 10,000+ images to simple extract RGB and Grey values from to feed into a CNN because it didn't seem efficient for this scale of a project. Thus, I researched and figured a cheap solution is to pad the RGB and Grey values with constant value with little deviation from the original set of values

Evaluating the Model

My model is a small conventional neural network that uses the rectified linear(ReLu) activation function for discrete classification of color values, hence the 3 separate CNNs for the respective color values. My model unfortunately has high loss between every epoch the CNN goes through. Since my computer does not have very good computational power, it takes a long time to complete one epoch, and skimping on the amount of neurons, neurons per layer, and hidden layers will decrease the necessary complexity of the CNN. The high amount of errors coming from the model most likely originates from the inflated color arrays received by padding the dataset and having only ReLu activation function for each hidden layer. However, I was told and looked up that for image classification that the ReLu activation function does not have the disappearing gradient error that other basic trigonometric functions have, and achieves a lesser cost function with every epoch. Realizing this, it may be my padded inputs that led the CNN achieve major overfitting.

Training the Model

I trained the model by using ReLu activation layers for my CNN to approach an optimal fit for prediction. I used the standard feed forward for predicting the color values from the data set, and backwards propagation to adjust the weights and calculate the loss from the mean of all the actual color values from all of the predicted values squared. Once again, I tried to determine convergence to create an optimal fit through using Relu calculation, and then use the derivative of the ReLu to adjust the weights for convergence. Unfortunately, if my loss keeps remaining high and constant through every epoch, it must mean that I'm heavily overfitting my data

Assessing the Final Project

Overall, my final project is a complete disaster. My predicted image is always black, which means that either my dataset is giving the wrong training values due to the cheap solution of padding, or that my activation layers aren't really "learning" and keep repeating the same errors, leading to severe overfitting. I fear that if I sample images of varying distinction i.e not just environments but people, cities, space, the ocean etc., then I fear that my CNN will face an even more difficult time fitting. Perhaps I needed to combine multiple calculations (Sigmoid, Tanh, log) and create a variety of activation layers. However, a big drawback to this is increase unnecessary complexity of the CNN of rather simple image classification. Also, I kept getting memory errors whenever I implemented a complex activation function equation, so maybe if I had a better computer and understanding of coding a CNN, I could achieve a quality colorization. Furthermore, I should've not padded the datasets, but simply taken the time to

acquire more images, or use a smaller dataset (500+) images from the internet. However, I really wanted to experiment and create my own training data to prove to myself that I can create a functional CNN. Even though my accuracy and precision of the model, theoretically, be drastically off of an industry standard of image classification such as R-CNN, it would be more resourceful to my knowledge if I learn hands on. I was also unsure of what I can and cannot use when it came to datasets for this project.

Doing this project by myself was a giant mistake. This is the hardest project I've ever done.