Advanced Machine Learning Project - DS-UA 301

Colorizing Black & White Images Using CNNs by Eli Bolker & Sabina Prochowski

A description of the project

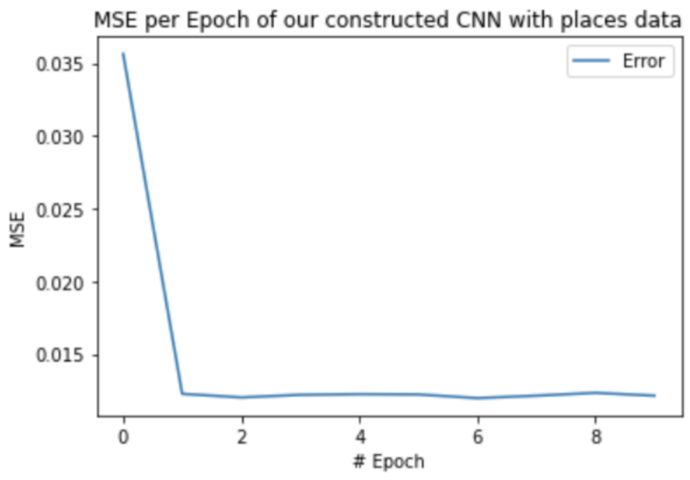
Our project evaluates the results produced by the black and white image colorization utilizing a basic CNN model vs. Resnet18 model. The goal of our assignment was to generate a colored image from a grayscale input image. This poses a multimodal issue, where a B&W can produce many plausible colored images. Consequently, for decades, colorizing images relied heavily on human input. This is because a user might know the color schemes that fit a particular photo given its setting and time period, but a neural network is not sophisticated enough to comprehend history and culture like we can.

Results (including charts/tables) and your observations

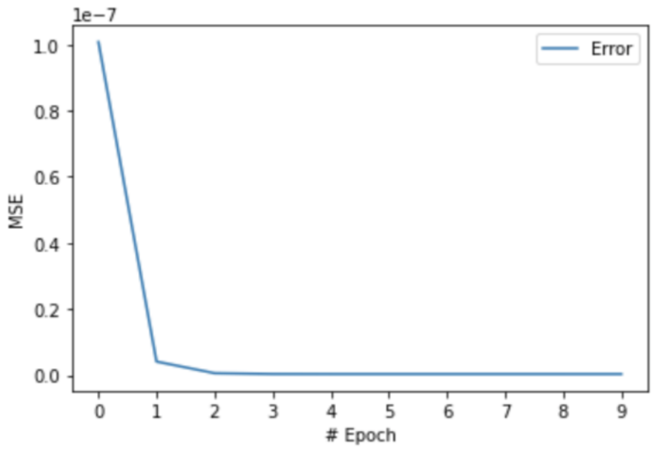
We trained the networks and converted the training images to Lab Colorspace, which quantifies the color via an independent color space. The Lab Colorspace uses three channels (L, a, and b) where the L channel takes in the grayscale input and the network predicts a and b. Those channels are combined with the L channel, which is then converted to RGB, producing the predicted colorized image.

For our CNN implementation, we follow the structure of https://emilwallner.medium.com/colorize-b-w-photos-with-a-100-line-neural-network-53d9b4449f8d. Initially, we used CNN architecture directly from the above link for our initial run. For our second run, we optimize several components: standardizing the train data, different optimization measures, different batch processing, different dataset than the site's dataset, and different processing of data. We then plot the loss differences over the epoch runs.

Below, you can see the following graph (refer to our presentation for further explanation):

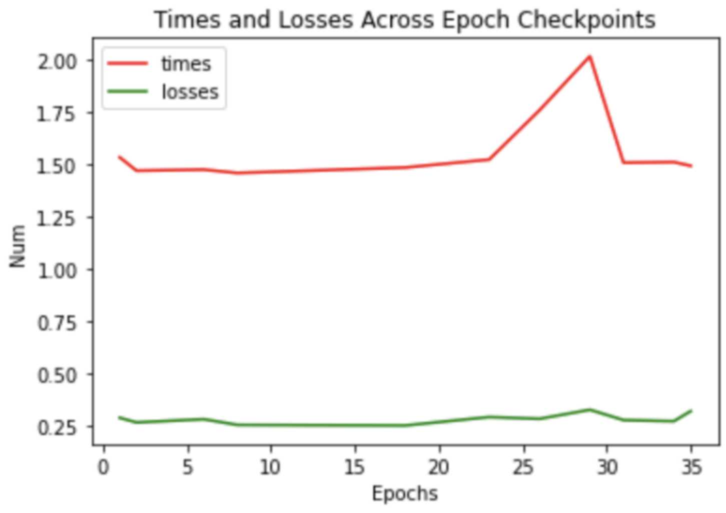


Before CNN Optimization



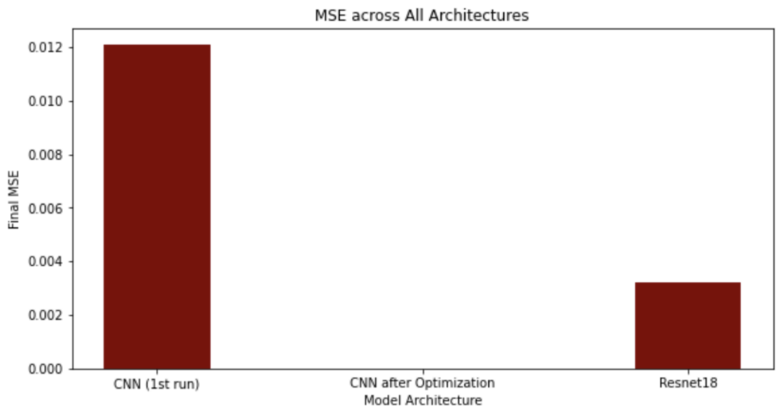
After CNN Optimization

As for the Resnet-18 implementation, we use the code from the following link: https://lukemelas.github.io/image-colorization.html. We attempted to hand implement the Resnet-18 model by hand, but ran into some technical issues - refer to our code, so we instead use the Pytorch's default Resnet18 model. However, we still run our own analyses of some checkpointed models (with the best losses) and their respective losses and time. Refer to the graph below.



Resnet18 Analysis

Final Comparison Analysis: Notice that CNN after optimization has the lowest MSE, yet is not the most subjectively realistic. The Resnet18 has the most realistic results. MSE and all other error measurements for CNN are imperfect, and do not necessarily represent ground truth in the way that they do in other NNs. Refer to the bar graph of these results below.



A description of the repository and code structure

- readME
- Powerpoint Folder
    - Stores the Adv ML Project - Presentation slides
- Code Folder
    - Stores the original CNN ipynb file (OriginalCNN ipynb file) of the original CNN implementation pre optimization measures
    - Stores the CNN (EditedCNN ipynb file) of the implementation post optimization
    - Stores the (EditedResnet ipynb) Resnet18 implementation
- PDF Folder
    - Contains all of the code files in PDF format

How To Run the Code:

Git clone the repository and open the ipynb files (no py files). We also include the PDF forms for easier reference. Please let us know in case of any issues.

Also, note that Resnet18 ipynb will take approximately ~9 hours to complete. The CNN implementations take approximately ~15 minutes to run, depending on your RAM storage and GPU availability. For this project, we used high RAM (30+ GB) and Google Colab Plus+. If these resources are not available for you, the sessions will crash.