Project Part 2: Neural Networks & Computer Graphics

CAP 5404 Deep Learning for Computer Graphics
Dr. Corey Toler-Franklin
Team Members: Preston Goren, Hunter Livesay

Implementation

Dataset Processing

We trained our model on the Georgia Tech Face Database as described in the instructions. We load the dataset directly from the link provided in the instructions(http://www.anefian.com/research/face_reco.htm) rather than from the course files to enable our image processing to happen totally in the cloud, without the need to upload the dataset. The images are resized to 128x128 via bilinear interpolation, and normalized between (0,1). We decided on the 128x128 size after experimenting with various sizes. Higher sizes ran into RAM usage limitations, and lower sizes made the resultant images look extremely grainy.

We perform 4 different augmentations on the data: zoom(crop), horizontal shift, vertical shift, and rgb scaling. These are combined together in various ways for a total of 9 augmented images for each image. After the training test split, there are 6075 augmented images.

A 74/16/20 train, validation, test split is used, and augmented images are used in the training set for our models but not the testing set as described in the project specifications. The augmented rgb images, l channels, a channels, and b channels are all stored and zipped separately for submission per project requirements.

Regressor

The l, a, and b channels are all normalized to a range of 0 to 1. From experimentation with converting RGB images to LAB, we found the maximum and minimum values to be 100, 184.439, and 202.345 for the L, A, and B channels respectively. For the faces dataset we are training on this means that all of our values for the A, and B channels lie between .214 and .909. While this means that parts of the normalized color space are unused, which would normally be a bad thing, we decided on this normalized scale so that when applied to different datasets it would generalize better, as a different dataset would likely use different portions of the spectrum.

Our regressor is fed as input a grayscale image (the $L*$ channel, scaled to the range [0,1]) and predicts the mean $a*$ and $b*$ channel values (ignoring pixel location). Our model is made up of 3 spatial convolutional layers, with a (3,3) kernel size, and (2,2) strides. Each layer has 3 filters, and uses relu activation. The final layer is a 2 width dense layer using sigmoid activation. It is trained using the adam optimizer, and mean squared error(MSE) for the loss function. On the test dataset it achieves a MSE loss of .000395.

Colorization

Like the regressor, the colorizing model normalizes the L, A, and B channels in the same way. The x data is the luminance channels of the augmented images, and the y data is the A and B channels as a 128x128x2 nd matrix.

We tried several different models, but ultimately decided on the following architecture. Each convolutional layer uses ReLU activation and a kernel_size of (3,3). Initially there are three downsampling layers with strides of (2,2), which are then followed by 3 filter layers with strides of 1, this is then followed by 3 upsampling layers. Each layer has 64 filters, except for the last which only has 2(representing the 2 output channels). We use the adam optimizer, have a batch size of 20, and train for 10 epochs. Note that we tried inserting batch normalization layers between each layer, but this resulted in a significant increase in loss for next to no increase in training speed. A single batch normalization layer at the beginning proved to be most efficient. We also decided to use mean absolute error(MAE) over mean square error. Experimentation with both showed that MSE generated qualitatively worse images.

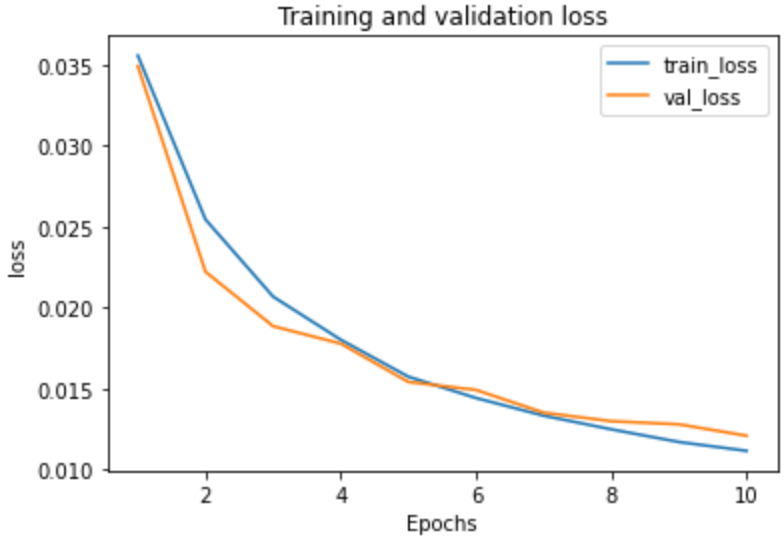
The above model was not the model that had the best quantitative performance. Once we switched to using GPU for training, we tried training much larger models. We were interested in seeing how adjusting the number of filters would affect the performance on the test dataset. Our hypothesis that at a certain point adding more filters would improve performance on the train dataset, but worsen on the test dataset due to overfitting.

Num Filters	MAE Loss on Test Dataset
4	.0292
8	.0230
16	.0198
32	.0182
64	.0165
128	.0145
256	.0135

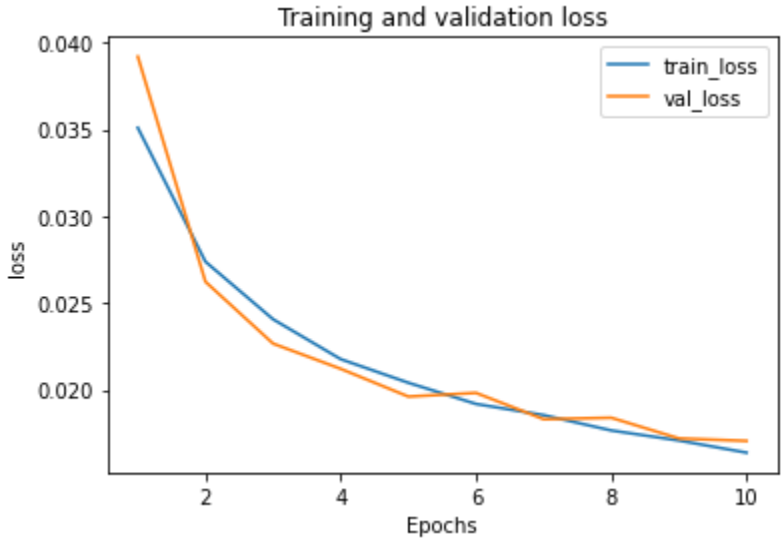
What we found was that increasing the number of filters always improved performance on the test dataset, although logarithmically. Exponential increase in the number of filters, resulted in linear decreases in loss. We suspect this is because the training dataset images are all relatively similar, although we add significant augmentation they are all in the same color space and feature similar images(a person and a bookcase). Thus there isn't significant enough difference between the training dataset and the test dataset for overfitting to occur. We say that the 64 filter model is the best, because it is the one that had the best results when evaluated on the NCD dataset.

The 256 filter model was trained on a Tesla T4 and took 8.6 minutes.

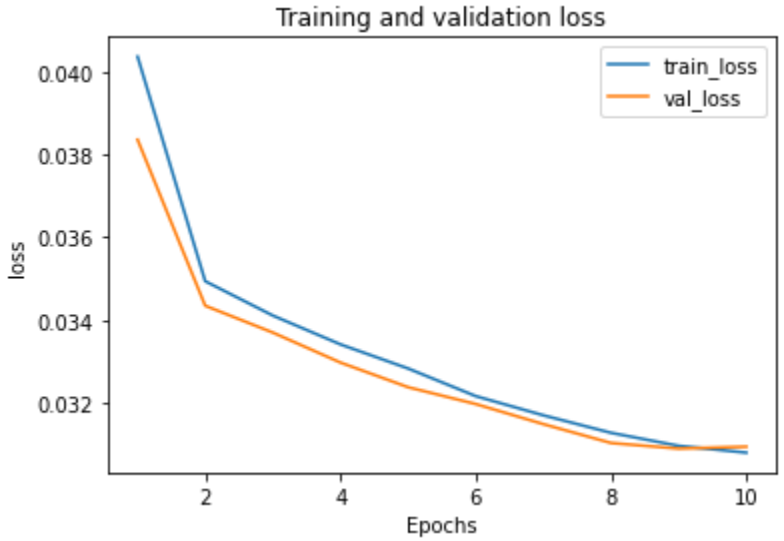
We also noticed that the higher filter models took longer to converge. Although all models were only trained for 10 epochs, the higher filter models didn't appear to be converging by the 10th epoch.



256 filter model



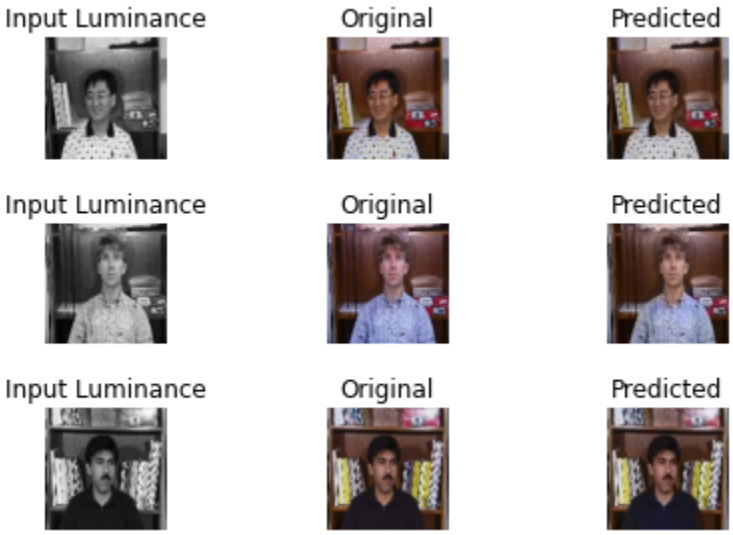
64 filter model



8 filter model

Colorization Results

The model overall performed very well on the test dataset, both quantitatively and qualitatively. Interestingly though, it consistently had difficulty coloring in the trop corners of the image. We suspect this is because the augmentation made the corners the most variable, and thus most difficult to learn.



Predictions of 64 filter model

In the second row you can see that the model had failed to correctly predict the shirt color, making it white instead of blue.

Transfer Learning

Our model had a MAE of .059 on the NCD dataset. It is also clear qualitatively that the model didn't transfer well.

In general the model tended to color the images brown. This is likely because the faces dataset mostly has brown colors because of the bookcase background. Having a more general dataset

that included a wider color palette would have allowed it to transfer to this task better. The higher filter models had no improvement both quantitatively and qualitatively.

Instructions for Running Programs

Here is the link to the Google Colab notebook used
https://colab.research.google.com/drive/1fcfTGwqTmin7r0ouDsmyox6KbcDWbcfG?usp=sharing
All training was run in a High-RAM environment. We have added code blocks that delete variables used during data augmentation, so that it can be run on a low-ram environment.

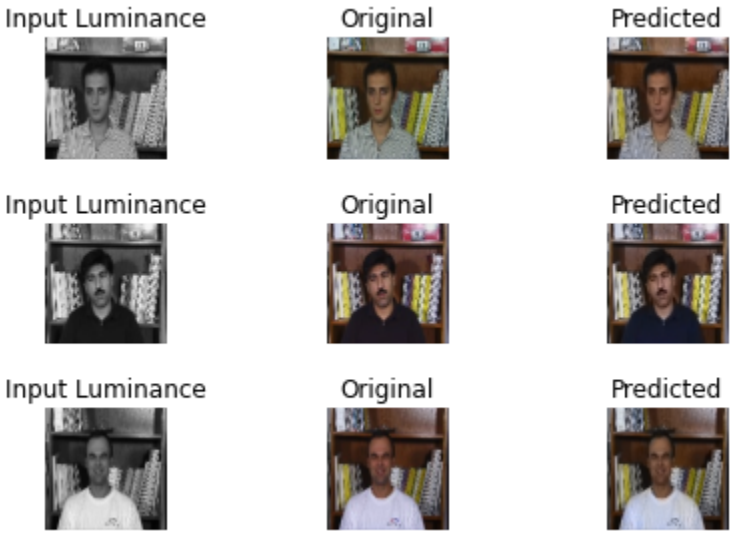
All of the cells can then be run in order. Downloading the faces dataset is relatively quick since it is run on Google's servers. The part on transfer learning does require uploading the datasets, we couldn't find an easy way to download these on the cloud. Thus the NCDataset, and ColorfulOriginal zip files **must** be uploaded to the runtime before it can be run.

Evaluation Results for Extra Credit (optional)

We decided to use the Tensorflow Keras api due to familiarity with it from previous projects. This meant that using GPUs for training was also very easy, since the Keras API takes care of most of the backend for you.

Tanh activation

We tried training the model with Tanh activation, using 65 filters per conv layer. Our input data was normalized between -1 and 1. The model had a MAE loss of 0.017, which was slightly higher than its loss when using sigmoid, but not significantly so.



Predicted images using tanh activation

Bugs, Difficulties (optional)

We initially had issues with converting the generated LAB images back to RGB. The issue ended up being that the conversion process assumes the LAB images are UINT8 when in our case they were float32.

Overfitting concerns

The model almost always was able to predict the color of the books in the background. This shouldn’t be possible given that there are many different colors of books. The fact that it is so accurate means that it memorized all of the orientations of books in the training dataset. If the application of this model was to color in pictures of people in front of bookcases it would perform very well, however it likely would fail to generalize to pictures of people in different backgrounds. More training data is required.