Project 2 Report

Sahil Agarwal, Audrey DeHoog, Michael Mezzina
University of Florida, sahilagarwal@ufl.edu, adehoog@ufl.edu, michaelmezzina@ufl.edu

***Abstract* - In this project, A Convolutional Neural Network was constructed to colorize Images of Faces, taken from the Georgia Tech Face Database.**

INTRODUCTION

This project sought to recolor a dataset of images taken of 50 people over two or three sessions. The images provided were pre cropped to include the faces of the subjects in the photo.

The original images occupied the RGB colorspace, where each pixel is assigned a separate Red, Blue, and Green value ranging from 0 to 255. In the context of colorization, it was found to be helpful to transform the images to the LAB colorspace. Where the three channels L, a*, and b* represent the lightness, and chrominance values respectively.

A convolutional Neural Network was chosen as the model through which colorization was to be achieved. This is due to the advantageous approach that CNN's take to process image data.

IMPLEMENTATION

For Data Preprocessing, the images were augmented in the following manner. All the images were resized to a 128x128 resolution. The images were then randomly cropped, horizontally flipped, and had their RGB values rescaled. These transforms were applied randomly, and were used to generate a total dataset 10x larger than the given dataset. For the regressor model, the two a* and b* channels of the LAB were averaged.

For the implementation of the models themselves, downsampling and upsampling blocks were used.

The CNN regressor used convolutional layers that perform a convolution with a 3x3 filter, and a 2x2 strid. This effectively halves the size of each input to the downsampling layer. Seven total layers were used, resulting in a single output. Three feature maps were used for all hidden layers. 2d batch normalization was performed after every convolution and the ReLU activation function was used.

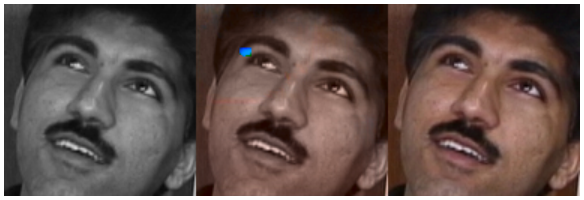
For the colorizer model, the first five blocks used were also downsampling blocks, with a matching stride and kernel size compared to the regressor. The number of feature maps varied however, doubling until it reached 48.

The next five layers in the model were "upsampling blocks" each block made use of a Nearest Neighbors upsampling layer, in conjunction with a convolutional layer with a 3x3 kernel size and a stride of 1. Batch normalization was performed after the aforementioned upsampling, followed by the ReLu activation function. The only variance from this architecture occurred in the final output layer, which made use of the Sigmoid activation function.

Both of our networks used the Adam optimizer for training, with a learning rate of 0.0001 and a mini batch size of 10. For the regression network, we used 2 epochs, and the colorizor was trained over 10 epochs.

The intuition behind our stoppage of training involved a calculation of validation losses at each epoch. Once the validation loss failed to drop, or began to rise, we stopped training.

RESULTS



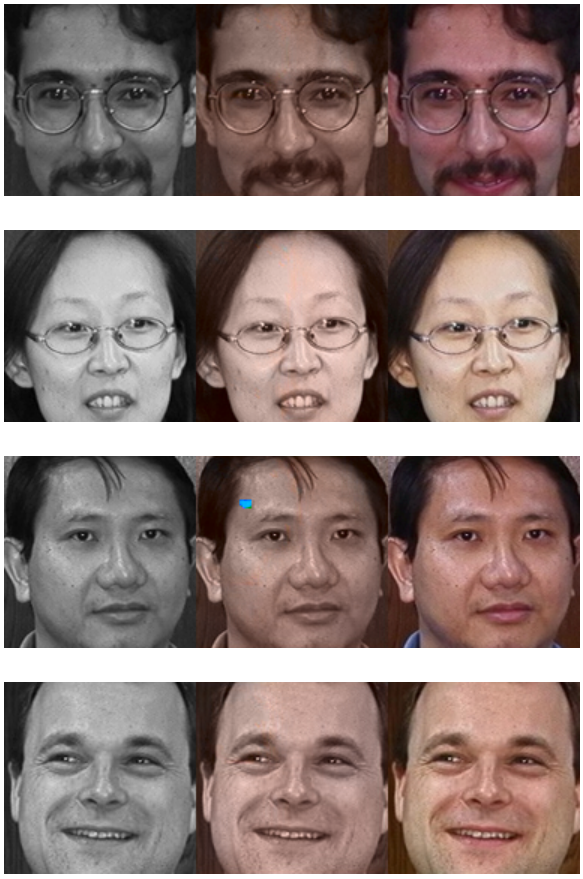


Figure 1: Left, luminance channel. Middle, image colorized with Colorizer network. Right, original image

	Loss (MSE)	CPU Time(s)	GPU Time(s)
Colorizer	0.0005	989	402
Regressor	0.0001	62	71

Table 1: Model Error and training times. GPU: NVIDIA GeForce 1650 (1024 cuda cores). CPU: AMD Ryzen 5 3550H

DISCUSSION

The images produced by our CNN were extremely close to the original images. After about 5 training epochs, we were getting MSE values in the range of 0.0005.

Since the images were mostly of faces, we noticed that our images tended to be tinted slightly red. This can be attributed to the predicting nature of a CNN. This is likely due to the overwhelming warm tone in the training data. An interesting point to notice however, was that the CNN tended to recolor irises as brown. This shows that the feature maps of the CNN abstracted high level features such as the eyes, and specifically the irises. It then colored them as brown due to the colors' prevalence.

A bug which we encountered found that some images would occasionally produce blue outlines or small areas of blue. This result may be caused by a saturation of the blue channel activations, in conjunction with associated feature maps. It seems that this phenomenon occurred mostly in the forehead region of the faces, where the likelihood of strong reflections was high.

A proposed solution to this problem may be to remove reflections through preprocessing.

APPENDIX AND ADDITIONAL INFO

Instructions on How to Run the Program

Run all the commands below from the root directory of the project. A virtual environment is provided with the zip file, it contains all of the dependencies required for the project. All files for training will autodetect whether a cpu or gpu is available. They will run on the cpu by default, unless a gpu is available.

For ease of use, the provided pretrained model is configured to run on a cpu, and does not require a graphics card. If running the pretrained model on a device with a graphics card, please manually set the device to ‘cpu’. This configuration is found at the top of the ‘ColorPredictor.py’ file. This is the only file which would need to be manually edited to run on a cpu.

Viewing Colorized Images

To see the results of the colorizer model, run the command

```
python3 ./models/ColorPredictor.py
```

this will open an image that shows both the predicted and original image. Any key can be pressed to cycle through the images. This program uses a pretrained model.

Training the Colorizer Network

To train the colorizer network, use the command

```
python3 ./src/models/CNN_colorizer_Model.py
```

this will train the colorizer model and save it to the ./model directory. At the top of the model file, the training parameters can be changed.

Training the Regressor Network

To train the regressor network, use the command

```
python3 ./src/models/CNN_Regressor_Model.py
```

this will train the regressor and display the loss on the training and validation set of the completed model.

File Explanations

ColorPredictor.py

The color predictor file is used to view a random image that is colorized with the colorizer model. After the colorizer and regressor models have been trained, this program can be run to view the colorized images. Pressing any key iterates through the images.

make_dataset.py

This file constructs a Pytorch dataset, which loads each image as it is accessed, in a memory efficient manner. This file also augments the data and changes the data to the LAB dataspace.No user action is needed to run this file, it is called by CNN_colorizer_Model.py and CNN_Regressor_Model.py.

modules.py

Contains custom convolution layers used by the regressor and colorizer model. No user action is needed to run this file, it is called by CNN_colorizer_Model.py and CNN_Regressor_Model.py.

CNN_colorizer_Model.py

Contains the model used to colorize the images. After running, the program loads in the face images dataset and augments the data using make_dataset.py. It then trains the model for a set number of epochs.

CNN_Regressor_Model.py

Contains the regressor model that predicts mean chrominance of the a and b channel. After running, the program loads in the face images dataset and augments the data using make_dataset.py. It then trains the model for a set number of epochs.