Course Project Report
Part II. Neural Networks & Computer Graphics

Pranath Reddy Kumbam
University of Florida, Gainesville, FL 32611, USA

UFID: 8512-0977

In this project, we will be exploring the application of machine learning in computer graphics. Specifically, automatic image colorization. The first part of the project involves the development of a convolutional neural network (CNN)-based regression algorithm for the prediction of mean chrominance values, followed by a colorizer model that colourizes an input image just by taking in the luminance information. The second part of the project is based on a transfer learning approach to develop automatic colorization models for the NCD benchmark dataset.

I. INTRODUCTION

Image colorization is the process of estimating color information for a given grayscale image. In this project, we will be using deep learning algorithms to develop an automatic image colorization pipeline. The application of machine learning for image colorization is an active area of research due to its extensive use in applications such as image and video restoration and also colorization for film and animation.

We will be studying two datasets with LAB color space in this project, specifically, the Georgia Tech Face Dataset and the NCD dataset. In the first part of the project, we will be training and testing regression models for the prediction of mean chrominance values (a*, b*) for the Face dataset, followed by an autoencoder-based encoder-decoder model that colorizes images using their luminance values (L). For the second part of the project, we will be using a transfer learning approach to use the models trained on the Faces dataset to develop automatic image colorization models for the NCD dataset. We will be using several methods, such as image augmentation, hyperparameter tuning, and batch normalization, to optimize our implementation.

II. DATA PROCESSING AND IMAGE AUGMENTATION

The Georgia Tech Faces dataset used in the first part of this project consists of images of people and faces. The dataset contains images of 50 people taken in two or three sessions between January 6, 1999, and November 15, 1999, at the Center for Signal and Image Processing at the Georgia Institute of Technology. All people in the database are represented by 15-color JPEG images with cluttered backgrounds, taken at a resolution of 640x480 pixels. The average size of the faces in these images is 150x150 pixels. The pictures show frontal and/or tilted faces with different facial expressions, lighting conditions, and scales. We will be using the OpenCV and NumPy packages in Python to load, process, and augment the images. The images are resized to the dimensions of 128*128 and then divided into training and testing/validation sets using an 80:10:10 split. We then expand the size of the training set by a factor of 10 using image augmentations such as horizontal flips, random crops, and scaling, and we populate the data tensor with the augmented copies using a simple python for-loop. The images are then converted into the LAB color space and saved into separate directories and NumPy arrays which will be used for training and testing the models. We will be using a similar approach to process the NCD dataset.

III. REGRESSION (AND HYPERPARAMETER TUNING)

In this section, I will be discussing the implementation and results of the regression models that predict the mean chrominance values of a given input face sample. I have first trained a simple fully connected model with [8192, 4096, 1024, 256] neurons in each hidden layer for 100 epochs with the learning rate set to 1e-5. The training loss and results are presented in Figure 1 and Table 2 respectively. We then explore convolutional neural network (CNN)-based models. I

Epochs	LR=1e-3	LR=1e-4	LR=1e-5
100	0.000196	0.000195	0.1877
300	0.000183	0.000197	0.1553
500	0.000198	0.000196	0.14998

(a) CNN1

Epochs	LR=1e-3	LR=1e-4	LR=1e-5
100	0.0001594	0.000128	0.000138
300	0.000135	0.000128	0.0001218
500	0.0001313	0.000128	0.000108

(b) CNN2

Epochs	LR=1e-3	LR=1e-4	LR=1e-5
100	0.000114	0.000109	0.000142
300	0.000129	0.000109	0.000136
500	0.0001108	0.000138	0.000135

(c) CNN3

TABLE I: Hyperparameter Tuning, Aggregate MSE Validation Losses of (a*,b*)

followed the project description while designing the architecture of the model. The CNN regressor includes 7 convolution blocks with stride and padding adjusted in such a way that the spatial dimension of the image is reduced by a factor of two in each layer. We test three different models with a different set of feature maps. The first model (CNN1) is relatively simple and has three feature maps per hidden layer. The second model (CNN2) has [16, 32, 64, 128, 256, 256] feature maps in hidden layers and the third model (CNN3) has [32, 64, 128, 256, 256, 512] feature maps. I have used grid search for hyperparameter tuning by training and checking the validation loss on various values of learning rate (1e-3, 1e-4, 1e-5) and the number of epochs (100, 300, 500) at a fixed weight decay of 1e-5. The models were trained using the Adam optimizer, and all the results of the validation loss of the tests for hyperparameter tuning are compiled in Table 1. The input values were scaled by dividing by 255, so to analyse the results in the original range, one can simply multiply by 255. As we can see, the CNN2 model performs the best at a learning rate of 1e-5 and 500 epochs. I will be using this architecture and set of hyperparameters for training and testing the final regression model, whose training loss and results are shown in Figure 2 and Table 2 respectively. Additionally, for extra credit, I have also tested the CNN2 model with a Tanh loss, whose training loss and results are shown in Figure 3 and Table 2 respectively. For Tanh's implementation, the values are first scaled to the range [-1,1] before passing to the model and are scaled back while testing.

FIG. 1: Training Loss of Fully Connected regression Model

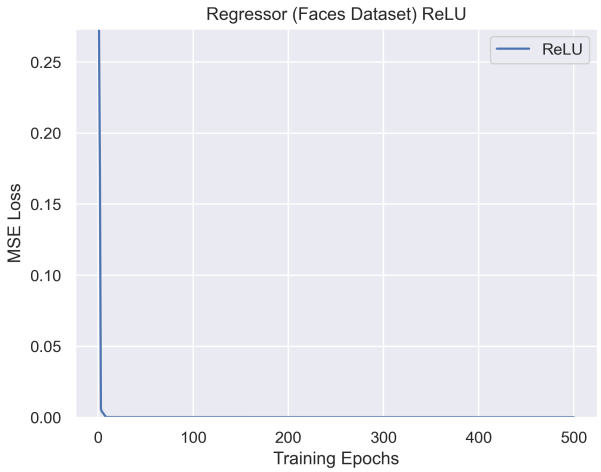

FIG. 2: Training Loss of CNN regression Model with ReLU

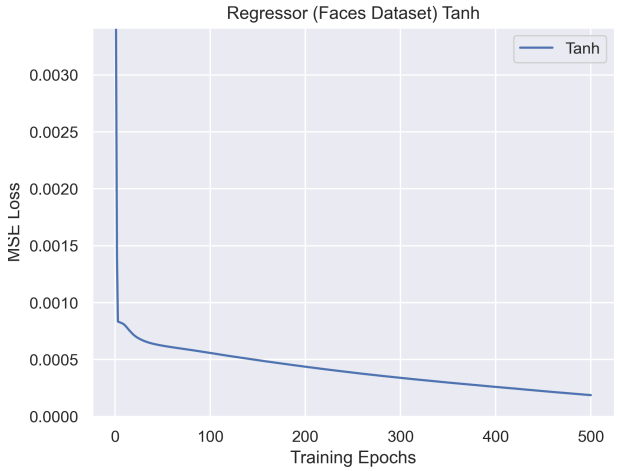

FIG. 3: Training Loss of CNN regression Model with Tanh

TABLE II: Regression MSE Results

Model	MSE (a*)	MSE (b*)	Learning Rate	Number of Train Epochs
Fully Connected Model	0.000468	0.000459	1e-5	100
CNN with ReLU	6.2262e-05	0.000172	1e-5	500
CNN with Tanh (Extra Credit)	4.40178e-05	0.000129	1e-5	500

IV. COLORIZATION (AND HYPERPARAMETER TUNING)

We will now be looking at developing encoder-decoder-based models for image colorization by including upsampling or deconvolution layers. The output spatial dimension of the image generated by the network is equal to the input spatial dimensions. As suggested in the project description, I have used five upsampling and downsampling blocks with batch normalization layers added before each spatial convolution layer. For hyper parameter tuning, the model architecture with respect to the number of filters remains the same, but I will be testing three activation functions, i.e., ReLU, Sigmoid, and Tanh for extra credit. Similar to before, I will be using grid

Epochs	LR=1e-3	LR=1e-4	LR=1e-5
100	34.5892	34.4374	34.46079
300	34.12332	34.6032	34.37195
500	34.57603	34.59363	34.30788

(a) ReLU

Epochs	LR=1e-3	LR=1e-4	LR=1e-5
100	34.424	34.66826	33.6417
300	34.5579	34.67858	34.6108
500	34.62515	34.435	34.4634

(b) Sigmoid

Epochs	LR=1e-3	LR=1e-4	LR=1e-5
100	35.4374	35.5301	32.1824
300	35.6492	35.3373	34.9099
500	35.75838	35.1595	34.8064

(c) Tanh (Extra Credit)

TABLE III: Hyperparameter Tuning, PSNE validation results

search for hyperparameter tuning by training and checking the validation loss on various values of learning rate (1e-3, 1e-4, 1e-5) and the number of epochs (100, 300, 500) at a fixed weight decay of 1e-5 . For Tanh's implementation, the values are first scaled to the range [-1,1] before passing to the model and are scaled back while testing. The models were trained using the Adam optimizer, and all the results of the validation loss of the tests for hyperparameter tuning are compiled in Table 3. As we can see, the best activation is Tanh, followed by Sigmoid, and the best set of learning rates and epochs is 10-3 and 500. The metric used for hyperparameter tuning is PSNR, and the final MSE and PSNR results are presented in Table 4. The training loss convergence of the models is shown below, and we also present some of the training and testing colorization samples.

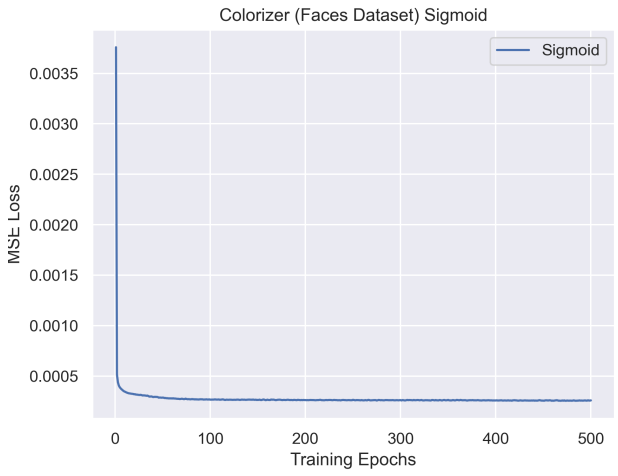

FIG. 4: Training Loss of Colorizer with Sigmoid Activation

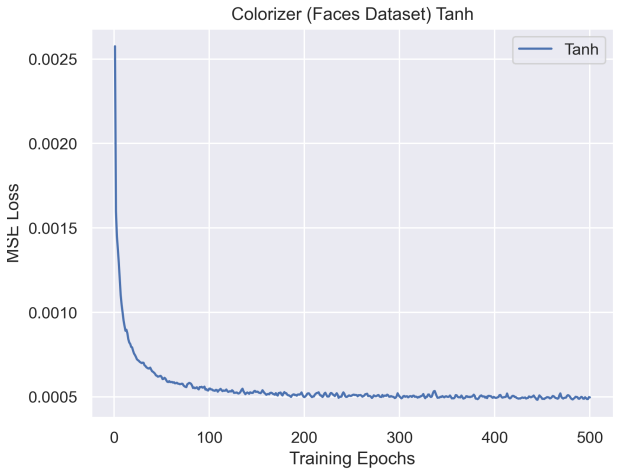

FIG. 5: Training Loss of Colorizer with Tanh Activation

FIG. 6: Example Colorized Train Samples (Sigmoid)

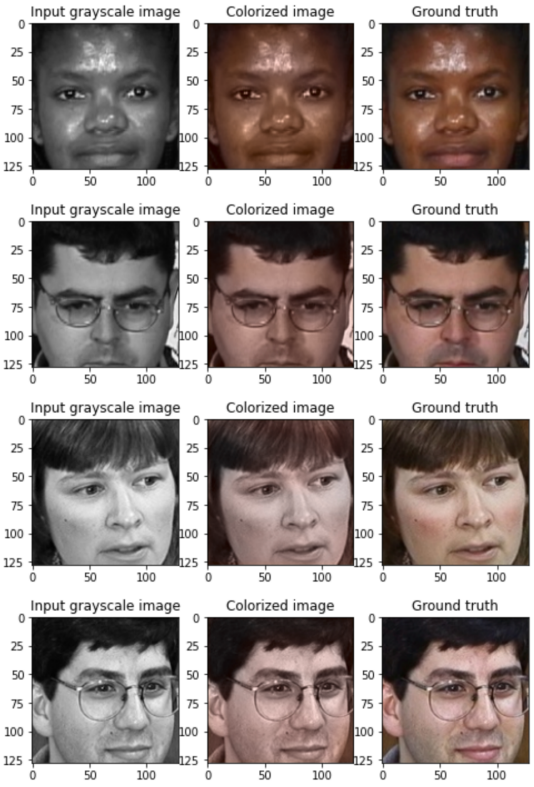

FIG. 7: Example Colorized Test Samples (Sigmoid)

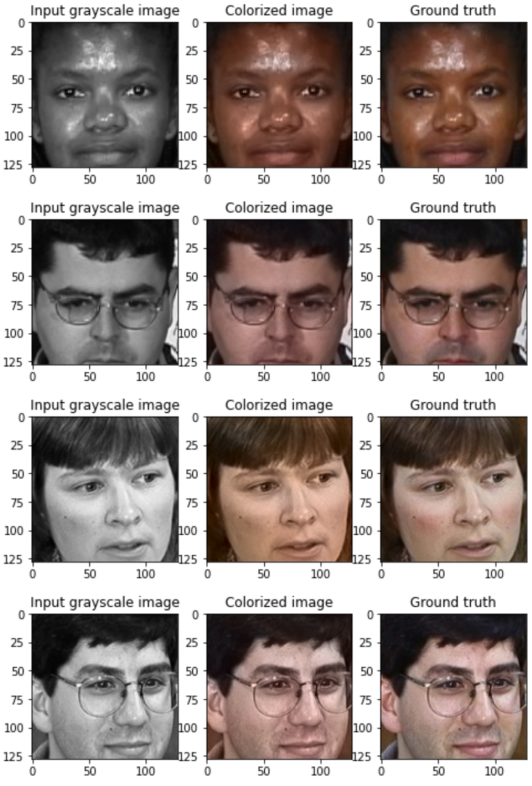

FIG. 8: Example Colorized Test Samples (Tanh)

TABLE IV: Colorization MSE and PSNR Results

Model	MSE (a*)	MSE (b*)	PSNR (a)	PSNR (b)
Colorizer with Sigmoid	0.000339	0.000406	34.69402	33.90814
Colorizer with Tanh (Extra Credit)	0.000222	0.000380	36.5217	34.1997

V. TRANSFER LEARNING

Next, we apply transfer learning by taking the best-trained models that we got for the Faces dataset and using them for the NCD dataset. Transfer learning is a machine learning technique where a model trained on one task is re-purposed for a second related task. We set the learning rate to 1e-3 and trained the regression and colorizer models that achieved the best scores on the Faces dataset for 1000 epochs on the NCD dataset. I have used ReLU and Tanh (Extra Credit) activation functions for regression and Sigmoid and Tanh (Extra Credit) activation functions for colorization. The training loss plots and the MSE results for regression are shown in Figures 9 and 10 and Table 5 respectively. For colorization, I have used the MSE, PSNR, and SSIM metrics as used in the reference paper. The training loss plots and the results for colorization are shown in Figures 11, 12 and Table 6 respectively. We also show some colorized training and testing samples.

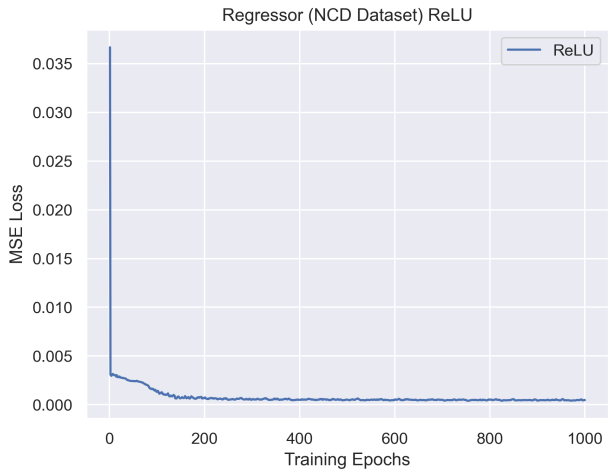

FIG. 9: Training Loss of regression Model with ReLU

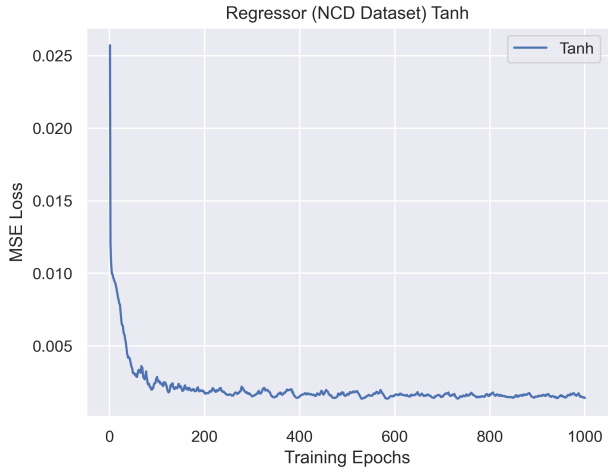

FIG. 10: Training Loss of regression Model with Tanh

TABLE V: Regression MSE Results for NCD

Model	MSE (a*)	MSE (b*)	Learning Rate	Number of Train Epochs
CNN regressor with ReLU	0.00134	0.00094	1e-3	1000
CNN regressor with Tanh (Extra Credit)	0.00164	0.00098	1e-3	1000

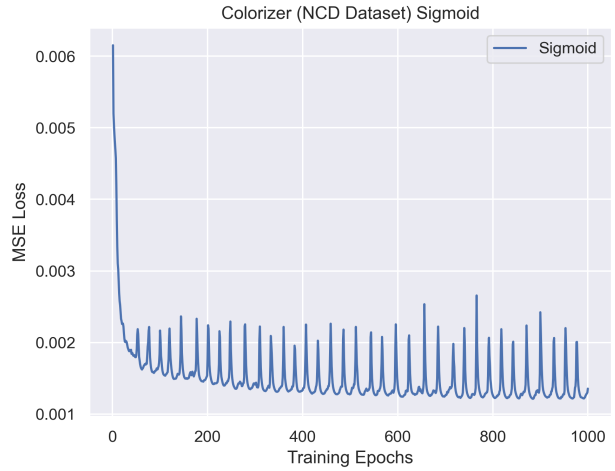

FIG. 11: Training Loss of Colorizer with Sigmoid Activation

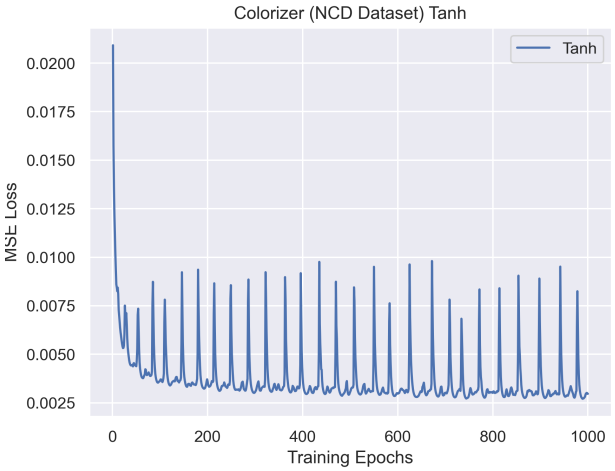

FIG. 12: Training Loss of Colorizer with Tanh Activation

TABLE VI: Colorization MSE, PSNR, and SSIM Results for NCD

Model	MSE (a*)	MSE (b*)	PSNR (a)	PSNR (b)	SSIM (a)	SSIM (b)
Colorizer with Sigmoid	0.005042	0.003354	22.97336	24.74351	0.706159	0.80465
Colorizer with Tanh (Extra Credit)	0.00332	0.00299	24.7877	25.2298	0.80334	0.83706

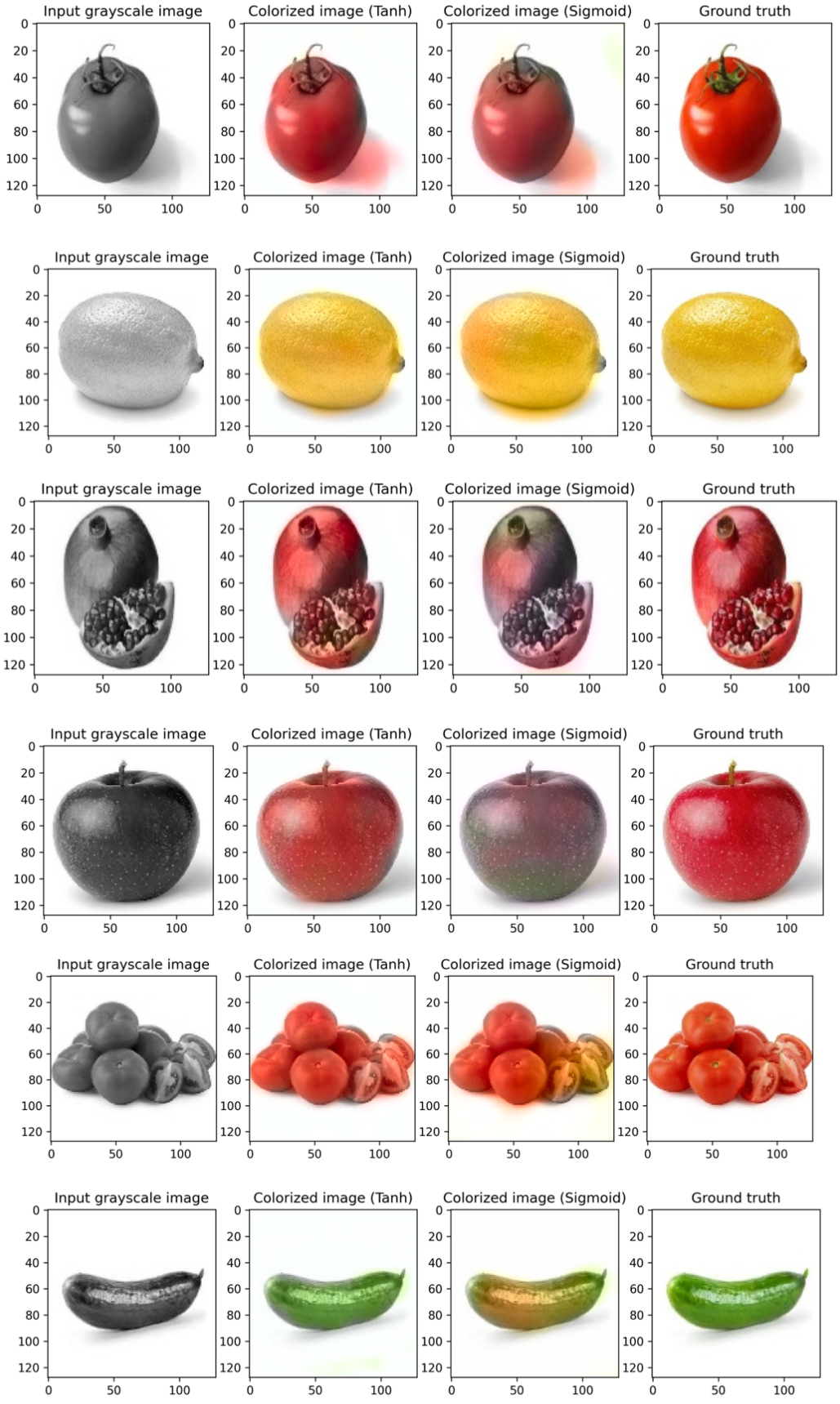

FIG. 13: Selected example Colorized NCD Test Samples (Tanh and Sigmoid)

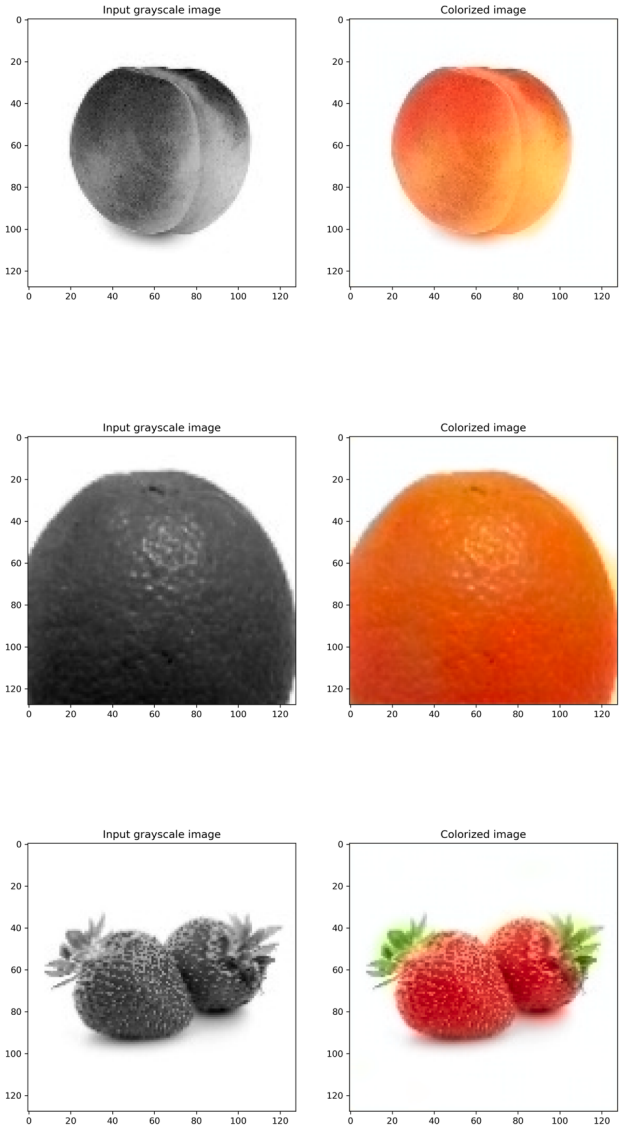

FIG. 14: Example Colorized NCD Train Samples

VI. EXTRA CREDIT

For one of the extra credit components, I have tested the training runtimes on CPU and GPU to compare the runtimes and calculate the speedup we get by moving the model to GPU. I calculated the speedup for a CNN regression model trained for 30 epochs and found the training time to be 8.5 times faster on GPU, which is quite significant. For the other extra credit component, I have already presented all the results with the Tanh activation function in the previous sections and in most cases, Tanh seems to be outperforming equivalent models with ReLU and Sigmoid activations, which is expected since the Tanh function is symmetric about the origin, which leads to easier learning and faster convergence.

VII. SUBMISSION DETAILS

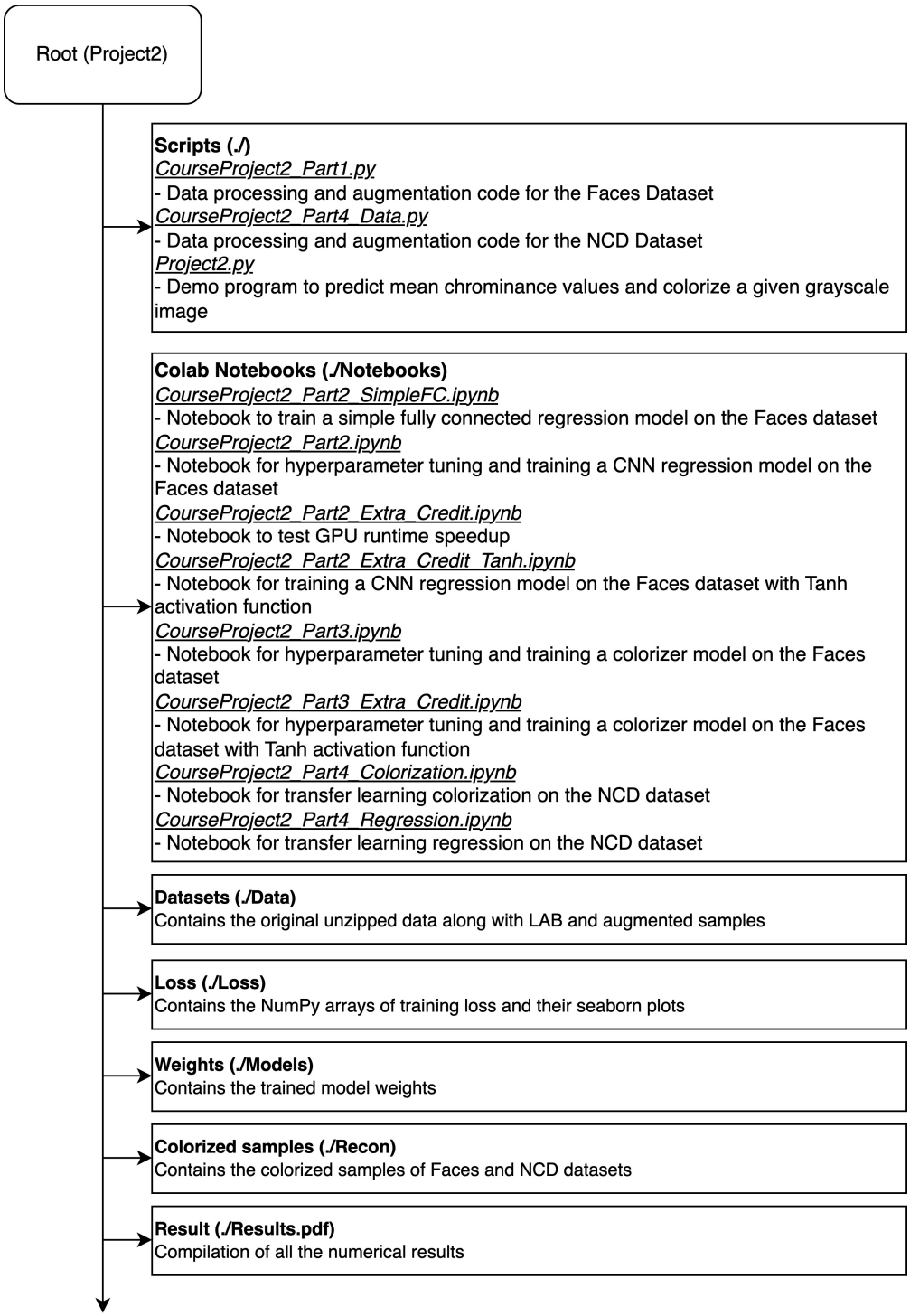

FIG. 15: File structure of the submission

Figure 15 shows the file structure of the submission and the details of each Python script and notebook. The notebooks are divided based on the types of tasks and models. As shown in the file structure, the model weights and data are stored in separate folders. The data folder contains the datasets in LAB format and also the augmented samples. The results are compiled into a pdf file and the colorized samples of Faces and NCD datasets can be found in the Recon folder. The demo script (Project2.py) is a program to predict mean chrominance values and colorize a given grayscale image. The test images can be placed in the './Test' folder and the program prompts the user to set the mode based on the type of test image (Faces or NCD). The folder containing the zipped project folder, file structure, and the project walkthrough video can be found at the following Google drive link:

Google Drive Submission

Regression

Simple Fully Connected Model	LR: 1e-5 Epochs: 100	Weight Decay: 1e-5	
Result (MSE)	a: 0.000468	b: 0.000459	

Hyperparameter Tuning:	Can be done by monitoring validation loss or a simple grid search, we will be performing grid search		

Three models: CNN1,2,3 (Different No Of Feature Maps)

Grid Search for Epochs: [100, 300, 500]	Learning Rates: [1e-3, 1e-4, 1e-5	Weight Decay: Fixed at 1e-5	Adam Optimizer, MSE Loss, ReLU Activation

CNN1 — Aggregate MSE Validation Losses of [a,b]

Epoch/LR	**1.00E-03**	**1.00E-04**	**1.00E-05**
100	0.000196	0.000195	0.1877
300	0.000183	0.000197	0.1553
500	0.000198	0.000196	0.14998

CNN2

Epoch/LR	**1.00E-03**	**1.00E-04**	**1.00E-05**
100	0.0001594	0.000128	0.000138
300	0.000135	0.000128	0.0001218
500	0.0001313	0.000128	0.000108

CNN3

Epoch/LR	**1.00E-03**	**1.00E-04**	**1.00E-05**
100	0.000114	0.000109	0.000142
300	0.000129	0.000109	0.000136
500	0.0001108	0.000138	0.000135

Best Model and hyperparameters:	Model: CNN2	LR: 1e-5 Epochs: 500 Weight Decay: 1e-5	This model and set of hyperparameters will be used for transfer learning
Result (MSE)	a: 6.2262e-05	b: 0.000172	
Testing best model (CNN2) with Tanh (Extra Credit)	a: 4.40178e-05	b: 0.000129	Model: CNN2 LR: 1e-5 Epochs: 500 Weight Decay: 1e-5

CPU vs GPU (Extra Credit)	Speedup: 8.5X for 30 epochs of training		

Colorization

Same Model Architecture but different activations

Grid Search for Epochs: [100, 300, 500]	Learning Rates: [1e-3, 1e-4, 1e-5	Weight Decay: Fixed at 1e-5	Adam Optimizer, MSE Loss

ReLU — PSNR

Epoch/LR	**1.00E-03**	**1.00E-04**	**1.00E-05**
100	34.5892	34.4374	34.46079
300	34.12332	34.6032	34.37195
500	34.57603	34.59363	34.30788

Sigmoid

Epoch/LR	**1.00E-03**	**1.00E-04**	**1.00E-05**
100	34.424	34.66826	33.6417
300	34.5579	34.67858	34.6108
500	34.62515	34.435	34.4634

Tanh (Extra Credit)

Epoch/LR	**1.00E-03**	**1.00E-04**	**1.00E-05**
100	35.4374	35.5301	32.1824
300	35.6492	35.3373	34.9099
500	35.75838	35.1595	34.8064

Best Model and hyperparameters:	Model: Sigmoid	LR: 1e-3 Epochs: 500 Weight Decay: 1e-5	This model and set of hyperparameters will be used for transfer learning
Result (PSNR)	a: 34.69402	b: 33.90814	
Result (MSE)	a: 0.000339	b: 0.000406	
Testing best model with Tanh (Extra Credit) (PSNR)	a: 36.5217	b: 34.1997	Model: Tanh LR: 1e-3 Epochs: 500 Weight Decay: 1e-5
Result (MSE)	a: 0.000222	b: 0.000380	

Transfer Learning

Epochs: 1000	LR: 1e-3	Weight Decay: 1e-5, Optimizer: Adam

Regression	**MSE Results**	
	a	**b**
ReLU	0.00134	0.00094
Tanh	0.00164	0.00098

Colorization

Colorization	**PSNR Results**	
	a	**b**
Sigmoid	22.97336	24.74351
Tanh	24.7877	25.2298

Colorization	**SSIM Results**	
	a	**b**
Sigmoid	0.706159	0.80465
Tanh	0.80334	0.83706

Colorization	**MSE Results**	
	a	**b**
Sigmoid	0.005042	0.003354
Tanh	0.00332	0.00299