Deep Colorization with CNNs

Shantanu Ghosh (4311-4360)   Kalpak Seal (8241-7219)

1 Steps to reproduce the output:

Download the file from the link **UF OneDrive** shared and unzip the file **DeepColor.zip** and execute the following command:
python3 main.py

A detailed description of the project is available at the **README.md** file. We have also included a **run_regressor.sh** file which will run the code in the hypergator.

2 Architecture

Our model consists of two networks - regressor and colorizer for calculating the mean of a and b channel, and to colorize the given input image respectively. Regressor network is responsible for downsampling the image and the output of the regressor network is fed into the colorizer network to upsample the same image. The configuration of regressor and colorizer are specified in table 1 and 2 respectively. The number of the channels of the feature maps are varied motivated by the Resnet 18 architecture.

The final layer of the colorizer is tested with both **sigmoid** (normal credit) and **tanh** (extra credit).

Layer	**Layer type**	**Depth**	**Kernel**	**Stride**	**Padding**
1	Convolution	32	4*4	2	1
2	BatchNorm	32	-	-	-
3	Convolution	64	4*4	2	1
4	BatchNorm	64	-	-	-
5	Convolution	128	4*4	2	1
6	BatchNorm	128	-	-	-
7	Convolution	256	4*4	2	1
8	BatchNorm	256	-	-	-
9	Convolution	256	4*4	2	1
10	BatchNorm	256	-	-	-
11	Convolution	512	4*4	2	1
12	BatchNorm	512	-	-	-

Table 1: Cofiguration of Regressor Network

3 Hyperparameters:

All the hyperparameters have been defined inside the **get_hyperparameters()** function within the **Utils** class in **utils.py** file. The parameters are defined as a dictionary and the function returns a list having the list of parameters respectively. Following this, during runtime, using **itertools.product**, we generate a cartesian product of the list of hyperparameters, the training has been done for each set of hyperparameters. If anyone wishes to add more hyperparameters, he / she can add it to **get_hyperparameters()** function within the **Utils** class in **utils.py** file. The hyperparameters, using which we have tested our code is given in table 3. The entire model has been developed using Pytorch framework.

4 Best Hyperparameters

Our experiments shows that the best results were obtained from the following set of hyperparameters for both sigmoid and tanh activation functions: **Learning rate: 0.001; Epoch: 100; Weight decay: 1e-5**.

Layer	Layer type	Depth	Kernel	Stride	Padding
1	ConvoTranspose2d	256	4*4	2	1
2	BatchNorm	256	-	-	-
3	ConvoTranspose2d	256	4*4	2	1
4	BatchNorm	256	-	-	-
5	ConvoTranspose2d	128	4*4	2	1
6	BatchNorm	128	-	-	-
7	ConvoTranspose2d	64	4*4	2	1
8	BatchNorm	64	-	-	-
9	ConvoTranspose2d	32	4*4	2	1
10	BatchNorm	32	-	-	-
11	ConvoTranspose2d	2	4*4	2	1

Table 2: Cofiguration of Colorizer Network

Parameter name	Value
Learning rate	[0.001, 0.0001, 0.1]
Epoch	[100, 200, 300]
Weight decay	[0, 1e-5, 1e-6]

Table 3: Hyperparameters (**extra credit**) for tanh and sigmoid activations

5 GPU

The code extracts the device at the start of execution using **torch.cuda.is_available()** and loads the tensors accordingly, leveraging the compute based on availability.

6 Loss Function plot

The loss function for the colorizer and the regressor network are plotted in the figure 1.

7 Output

7.1 Regressor

The regressor outputs two scalar values which are the average of the a and b channel for each of the images in the console.

7.2 Colorizer

The output of the colorizer network with colored images are stored in the respective **outputs_sigmoid** and **outputs_tanh** folders based on the activation function sigmoid and tanh respectively. The grayscale and colorized images are stored under the gray folder and color folder within **outputs_sigmoid** and **outputs_tanh** folders respectively.

8 Optional Credit - MSE of a channel and b channel

As mentioned in the question, we initially tested the model with a small number of feature maps as 3. As a part of this step, we varied the number of feature maps as 6 to 512 and tried to see the MSE between the true and predicted value of a and b channel. We found out the optimal number of feature maps as specified in table 1 and table 2 and the MSE computed as **0.000136** and **0.000117** for **sigmoid** and **tanh** respectively.

9 Optional Credit - tanh as activation function

For **normal credit**, we used sigmoid as the activation function for the final layer of the colorizer network and tanh for the **extra credit**. The output for the run with the optimal hyperparameter setting for **sigmoid** and **tanh** activation functions are shown in **Figure 2** and **Figure 3** respectively.

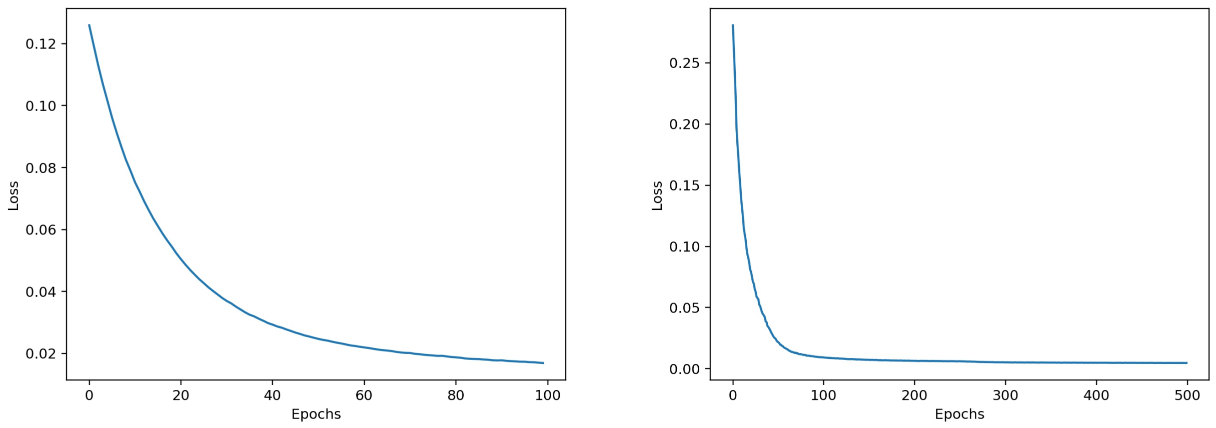

Figure 1: The plot of Colorizer(left) and Regressor(right) Network

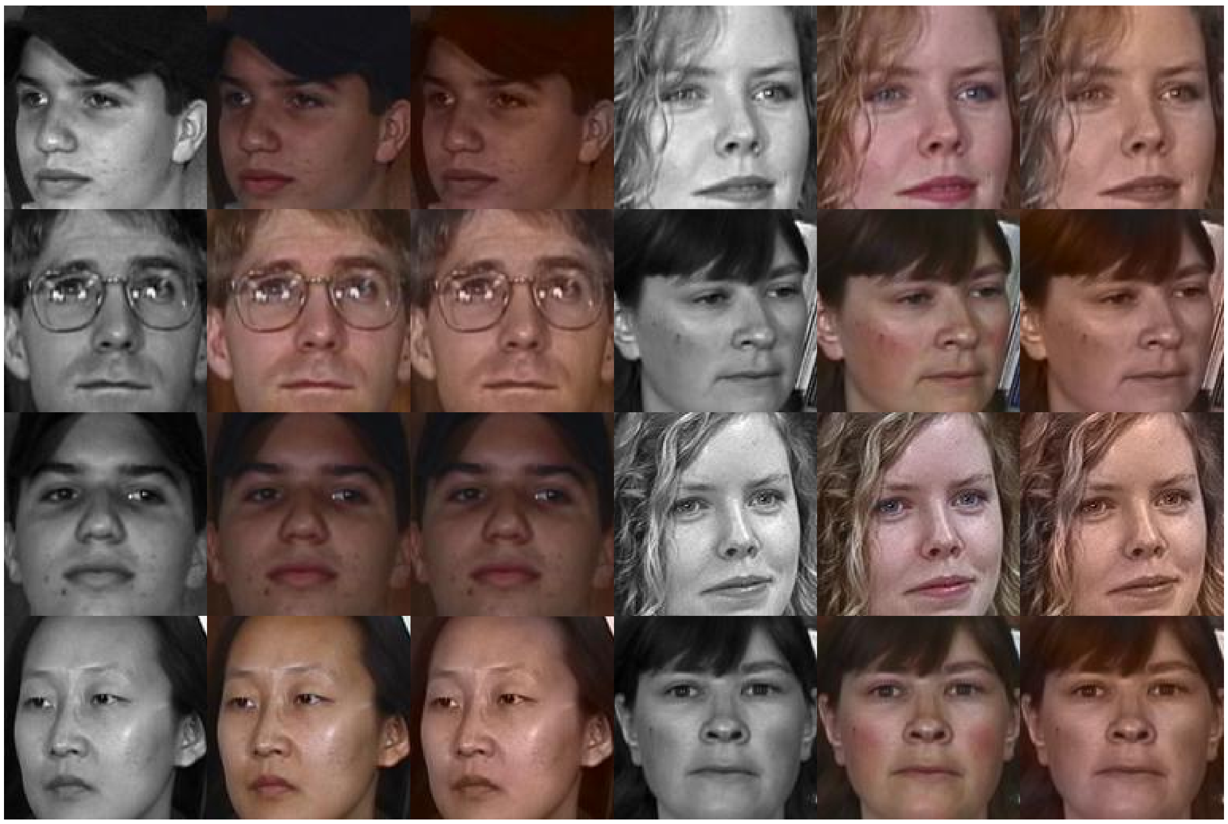

Figure 2: Generated images for the run with the optimal parameter setting with activation function as **sigmoid**. For every triplet, from left to right are gray scale, original and colored reconstructed images

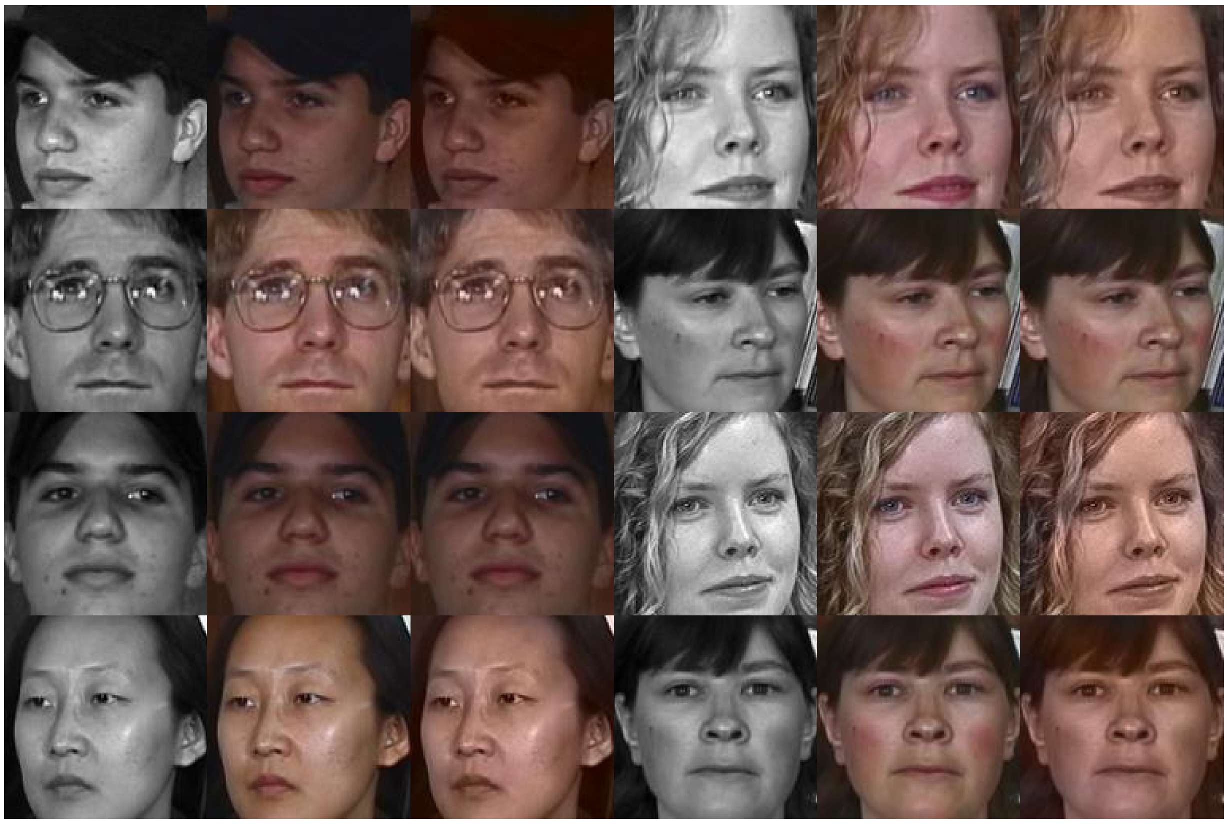

Figure 3: Generated images for the run with the optimal parameter setting with activation function as **tanh**. For every triplet, from left to right are gray scale, original and colored reconstructed images