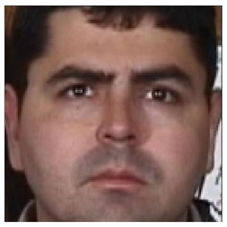
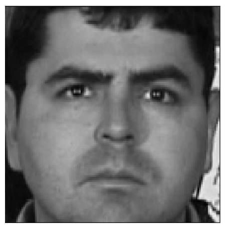
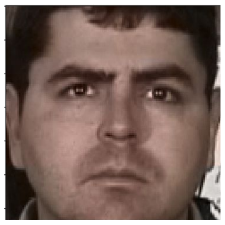
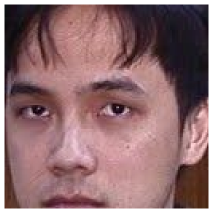
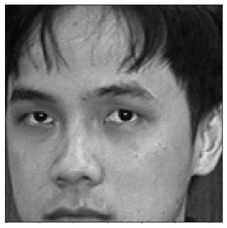
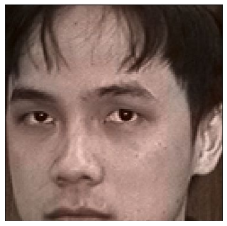
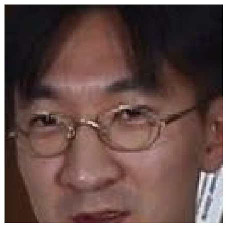
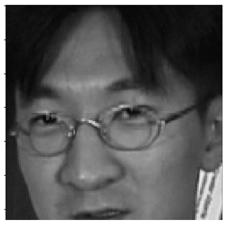
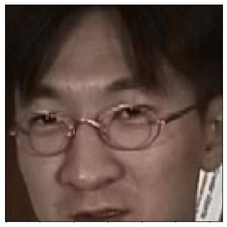
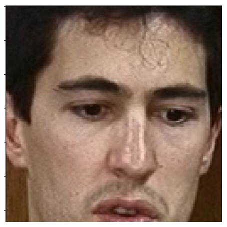
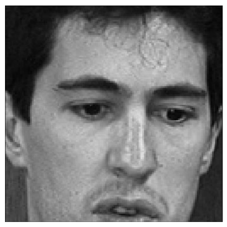
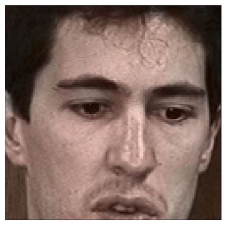
COP6930/4930 Project-2 Report

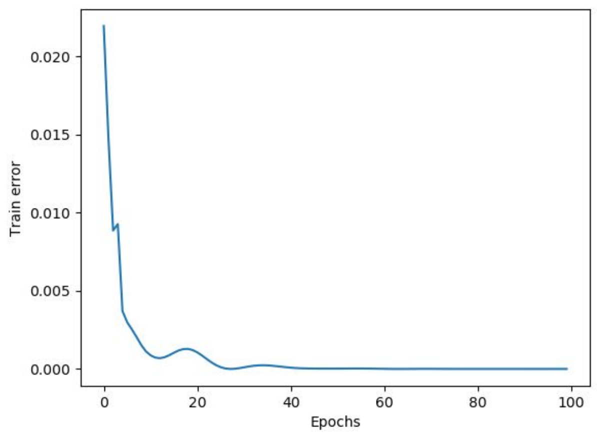
Kapish Yadav Banda(UFID: 98826143), Nickolas Kroeger(UFID: 5224-5139)

1. **Implementation Explanation**
   a) Read Data
   We read the data using skimage.io
   Divided the train test composition as 90% train and 10% test data.
   b) Augment Data
   Augmented the train data only in 3 ways - horizontal flip, center crop and scaling RGB values with a random number in [0,1].
   We Augmented them 3 times for each way and added them to the training data which made a total of 10 * 675 = 6750 training images and 75 test images.
   c) Convert to LAB color space
   The RGB to LAB conversion was done using
   `skimage.color.rgb2lab()`
   d) Build a Regressor:
   For the regressor we first split the LAB images into L, a* and b* channels.
   Then normalized L channel to [0,1], a* channel and b* channel to [-1,1]
   Now we got the average of a* and b* channels for our ground truth and saved it as a tensor of shape (N, 2, 128, 128).
   Model: We build the model with 7 Conv2D layers with appropriate kernels, stride and padding. Used Relu for input and all hidden layers but Tanh for the output layer.
   Trained model using MSELoss and Adam optimizer
   e) Colorize the image
   After splitting the image into LAB channels and then normalizing them, we constructed a model called ColorNet with 5 Conv2D downsampling layers and 5 ConvTranspose2D upsampling layers. Used LeakyRelu for all the input and hidden layers but used Tanh for the output layer.
   f) Batch Norm
   In the ColorNet Model we used `torch.nn.BatchNorm2d` before each activation layer to perform batch normalization.
   g) Training
   Trained on 6750 images with ColorNet, Adam optimizer, MSE Loss, L channel and a, b channels (all normalized) for 15 epochs
   h) GPU Computing
   Computed using a local GPU

2. Evaluation Results

Original **L Channel (Gray-scale)** **Predicted Channel**



MSE Test Set Loss for the Colorization network: 0.0021

The figure to the left is the training loss decreasing for the CNN Simple Regressor.

3. **Run Instructions**
    a) Regressor Model: Run the following command-
    ```
    python Proj2.py
    ```
    b) Colorization Model: Run the following command-
    ```
    python Proj2_Colorizer.py
    ```

4. **Bugs/ Difficulties**

We had difficulties with cv2's functions for converting between RGB and LAB spaces as well as displaying and reading in images. We opted for sklearn as it was easier to use - it gave less problems later on.

1st Extra Credit

We implemented a tanh layer on the output and scaled the a and b channels to be in the range [-1,1] and later mapped back to [-127,127].

2nd Extra Credit

We explored different model architectures by changing the input and output channels for each layer in the Conv2D and Upsampling layers. The models with these architectures are saved in the Model folder.
The **ColorNet.pt** model has all input and output channels as 3 except the activation function of Tanh where the outpul_channel was 2.
With **ColorNetNew.pt**, we experimented with differing channel sizes, where we went from 1 -> 8 -> 16 -> 32 -> 64 -> 32 -> 16 -> 8 -> 1.
We also did a larger architecture named **ColorNetHuge.pt** where we did 1 -> 8 -> 32 -> 64 -> 128 -> 512 -> 512 -> 128 -> 64 -> 32 -> 64 -> 8 -> 1.

3rd Extra Credit

Colorization can use a classification loss as shown by Zhang et al.'s paper "Colorful Image Colorization". They treat the colorization problem as multinomial classification by discretizing the a*b* space into bins of size 10. This allows for learning a mapping over possible colors at each pixel, and then they use a multinomial cross entropy loss. This approach has shown high-quality results for colorization, but its strength lies in its ability to model a distribution of color possibilities over a pixel. This is useful since an object can take on several possible colorization mappings.