Image Colorization

Name: Parth Shettiwar
Email: parthshettiwar@g.ucla.edu
Test: Samsung Research Neon test

Abstract

In this assignment, the problem of Image colorization was solved where we are given a grayscale image and output should be a RGB image. As we see, its an inverse problem and hence understanding the underlying distribution of dataset is important to make any prediction at first place.
The dataset consists of landscape images, and any network would try to understand the range of colours which an usual scenic image consists of (Blue sky, green grass etc.)

Implementation

The given folder consisted of 3 files: colorize_data.py, basic_model.py and train.py

- The assignment first asked to write a custom dataloader function in colorize_data.py.

This was done in a standard manner. The dataset sometimes had 4 channel or 1 channel image too other than prevalent 3 channel RGB images. These 4 and 1 channel images are filtered out on the fly while fetching an image from data loader.
In dataset, we are given RGB images. With these RGB images, grayscale image is first produced (to be fed into network). In addition to this, RGB images is converted into LAB image. This is done using rgb2lab function of skimage library. I also scale this output between 0 to 1. LAB expresses color as three values: L for perceptual lightness, and a and b for the four unique colors of human vision: red, green, blue, and yellow. In addition to this, groundtruth image is also returned (for comparison later)

- Then we had to implement a loss function and optimizer in train.py

For optimizer, adam optimizer was used, as its quite famously used in Deep learning (theoretically proved to be effective over other optimizers)
For loss function, I compared the “ab” channel of ground truth with output of network, which is again 2 channel (since the first channel is L and that is equal to grayscale, hence we can use it later to generate the final image)
Following were the attempts for loss function and final solution:

1) Standard MSE loss: Produces coloured images but were wrongly coloured on various scales. This is because, even if the pixel value deviates by little amount, still loss is less but this results in neighbouring pixels being of different colours
2) MSE + L1 loss: Introducing L1 loss, solves the above problem upto some extent, since now if it deviates, the loss would be incurred on higher scale than if it was squared. L1 loss ensures, that nearby pixels are almost of same colour intensity range.
3) MSE + L1 - SSIM: Introducing SSIM loss, does not help much since usually it is used an evaluation criteria and if we take negative of it to consider it as a loss, then network can learn in such way that total loss is less than 0 and that wont be effective for learning

Hence finally **MSE + L1 loss** was adopted as final Loss function

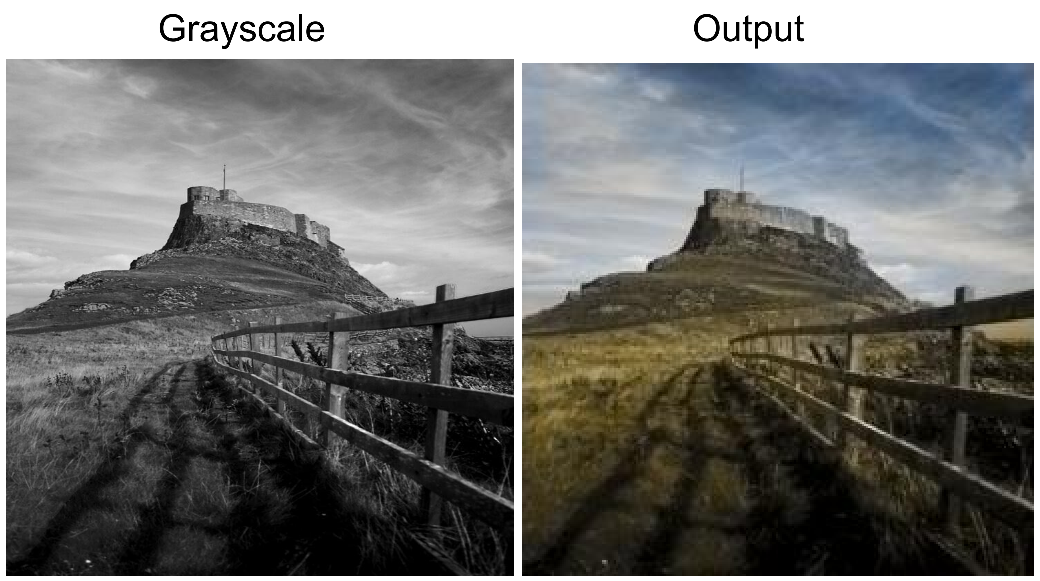
- Finally it is asked to complete the training and validation loop.

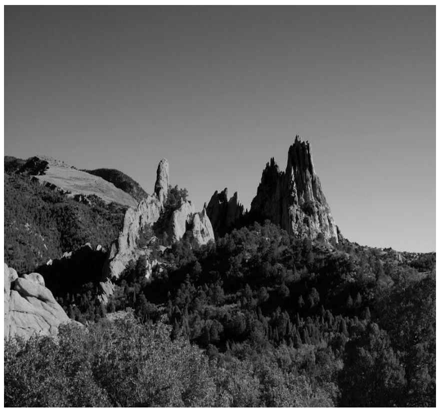
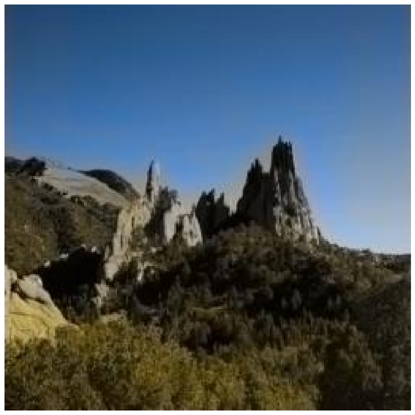
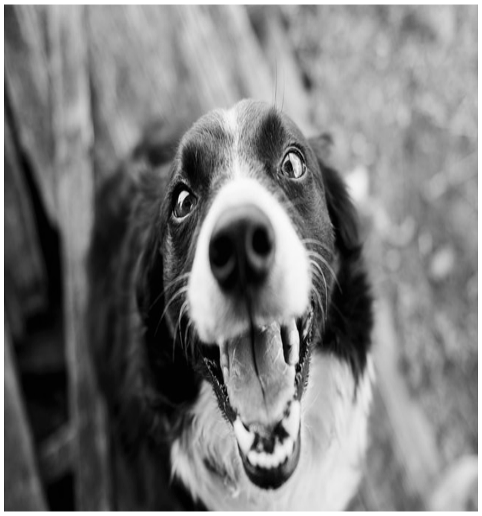
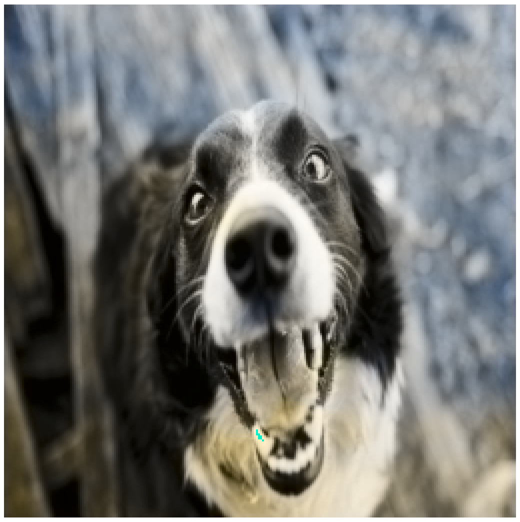
The final training is done on complete dataset and for experimental purposes 80:20 split was used for training vs validation. The number of epochs used is 10 and batch size of 16.
The whole training is done on Google Colab

- Inference script

In this, it is important to perform appropriate post processing steps to get the final image.
As described earlier, the network learns to predict the “ab” channel of the image. With this output, we first combine grayscale input with “ab” channel prediction to make it 3 channel. Now we first rescale the image (since previously we had used scaling) where L channel is multiplied by 100 and “ab” channels by 255 and subtracted by 128. (Inverse of operations used previously). This is now converted to rgb, using lab2rgb function of skimage library to generate the final coloured image.

Results



Interesting to observe is that, model gives good predictions on non-landscape images like the one above of dog where it produced fair enough coloured dog image. At same time, its not that accurate since training has been done only on landscape images. Output size in all images is 256 x 256.

Metric

Used the (MSE +L1) loss as metric for evaluating models performance
The final validation Loss goes down til 0.0009. The lower the loss, the better is models performance
This metric is valid here as MSE and L1 describes the direct relationship between predicted and groundtruth image.
We can also use SSIM and PSNR metrics
Average SSIM = 0.95
Average PSNR = 22.8

Both of these values are quite good (SSIM can max go till 1). I used these metrics as they are common computer vision image similarly metrics. SSIM measures the image similarity on window patches and PNSR is inversely proportional to MSE loss.

Additonal Tasks

To improve the models overall performance, I employed the following tricks:

1) Computer everything in LAB space rather than RGB space, which helps in better separation of channel spaces as they are independent and more related with Grayscale image
2) Used the (MSE+L1) loss after many variations of loss functions to get a good qualitative generated images
3) Changed the model output to 2 channel (the given network produced 3 channel output). Specifically training was done in a way that model should predict the “ab” channel of the groundtruth.

All of these have been detailed in previous sections.