CS 577 Final Project Report

**Colorization of Black and White Photos using Neural Networks**

By

Mayank Reddy Saddi

A20423747

**Problem Statement**

This project addresses the problem of generating an automized method to colorize black and white images. Previous solutions to this problem relied heavily on significant human input to hallucinate possible colors for sections of black and white images. This involved obsessive professionals spending months on photoshop to imagine appropriate colors for different segments in the image and filling in the segments with those colors. This process requires extensive research with just faces taking multiple shades of pink, blue and green to get an accurate color.

Although there have been many services which allow colorization most of these often require user input to assign colors accurately. This project aims to automize the process of colorizing greyscale images with realistic colors. In this project we use Convolutional Neural Networks to automatically colorize black and white images.

Image colorization assigns a color to each pixel of a target greyscale image. Colorization methods can be roughly divided into 2 categories –

1. Scribble based methods

These methods require significant efforts from the user to assign accurate colors to greyscale images. This is a very time-consuming method specially for images with fine scale structures.

1. Example based colorization

Example based colorization methods were later proposed to reduce the burden on users and automate the process. This method transfers color information from a similar reference image to target greyscale image. However, finding similar reference image becomes an obstacle. Thus, instead of finding entire images for reference target images are segmented into objects and then most similar image patch/pixel in in a reference image database and then transferring color information from the matched patch/pixel to the target patch/pixel.

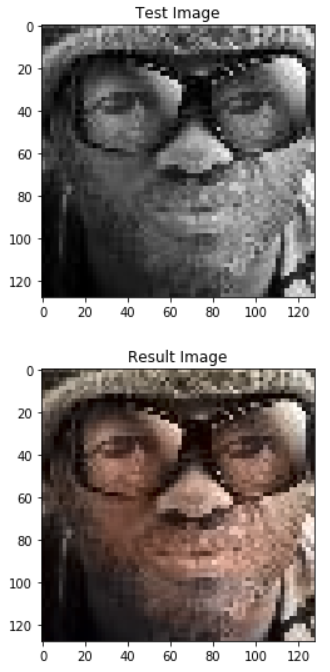
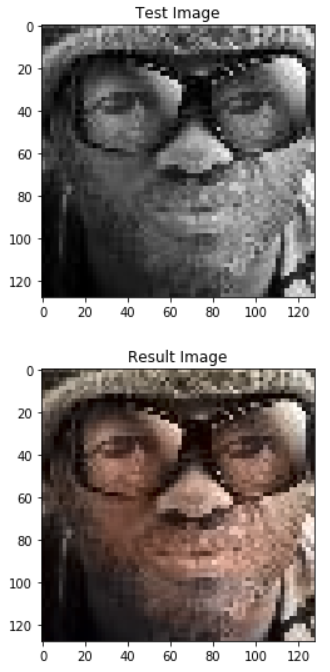
The goal of the project is to take a simple greyscale image as input and use a Convolutional Neural Network model trained on multiple images to generate a colorized output which is realistic. The produced output does not require to match the exact ground truth image but instead should be realistic enough to fool an average Joe.

**Implementation Details**

1. **Data Acquisition**
2. TimyImageNet 200

Initially to test out the basic architecture and functionality of CNN model. The model was trained on subset of images from TinyImageNet 200 dataset which consists of 200 classes of images with each having 500 images each of 64x64 pixels in dimensions.

Once the model was trained accordingly it was observed that the images in this dataset had very low resolution and the results looked very pixelated and unclear.

1. Labelled Faces in the Wild

Thus to obtain better results the model was again trained on Labelled Facec in the Wild (LFW) dataset consisting of more than 13000 images of people collected from the web.

Each of these image are 250x250 pixels in dimensions and allowed for better results.

1. CelebA Dataset

Like the Labelled faces dataset celebA dataset also contains several images of celebrities which is popularly used in computer vision and facial recognition projects. Since the subjects in most black and white images are people the model was trained on LFW and CelebA datasets to work well with human faces.

1. **Data Preprocessing**
2. Colorspace

All the images in our dataset are in RGB colorspace. First each of these images are converted to CIE Lab colorspace. Images are converted to this colorspace since in Lab. L channel represents Lightness which when extracted from the image gives us the greyscale version of the image.

The a and b channels represent the green-red and blue-yellow spectra of the image forming the images color.

With images in Lab format an image with L channel extracted is given as input and the model predicts a and b channels for it.

Thus the problem can be structed as supervised learning in which the predicted a and b channels are compared with the original values and frame a sort of regression model.

1. Reducing the size

Since the total dataset contained almost around 15000 images loading them as numpy arrays was very taxing job with the system’s limited memory of 12GB. Thus the images before being stored in numpy arrays each was reduced to a size of 128x128 pixels each.

1. Shuffling and Data Augmentation

To improve the performance of the model and generalize it better the images were shuffled using sklearn.shuffle before being split into train, validation and test subsets.

To allow for better genralization and prevent overfitting Data Augmentation is performed using keras.preprocessing.ImageDataGenerator to perform random flips, skews, rotations and zooms to images.

1. **Model generation**

Before the actual model generation severals approaches which have been implemented earlier were researched which used large Convolution Networks, Generative Adverssarial Networks (GAN), encoder, decoder and fusion layers.

Due to the limited resources of Computational Capacity of the system at hand (an Nvidia GTX 1050ti, 12GB RAM) and time constraint the model was generated using only Convolutional Neural Networks.

The model was built based on the one proposed my Zhang et al [1] with the following atchitecture:

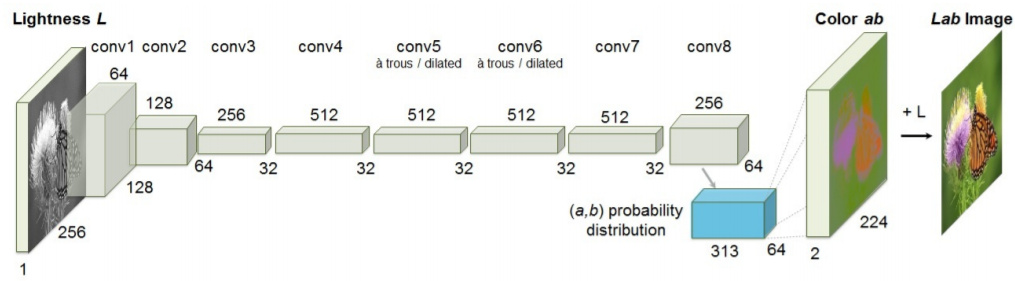


Figure Zhang model

Model Architecture:

Layer 1 Input 128x128

Layer 2 Conv2d 64 3x3 padding = same relu

Layer 3 Conv2d 64 3x3 padding = same strides = 2 relu

Layer 4 Conv2d 128 3x3 padding = same relu

Layer 5 Conv2d 128 3x3 padding = same strides = 2 relu

Layer 6 Conv2d 256 3x3 padding = same relu

Layer 7 Conv2d 256 3x3 padding = same strides = 2 relu

Layer 8 Conv2d 512 3x3 padding = same relu

Layer 9 Conv2d 256 3x3 padding = same relu

Layer 10 Conv2d 128 3x3 padding = same relu

Layer 11 Conv2d 64 3x3 padding = same relu

Layer 12 Conv2d 32 3x3 padding = same relu

Layer 13 Conv2d 2 3x3 padding = same tanh

Using relu in final layer restricts to only positive values model was not able to predict respective blue green channel spectrums.

To prevent overfitting dropout was implemented between hidden layers with a rate of 0.3. Trying out different rates from 0.2 to 0.5, 0.3 produced best results.

For loss function Mean Squared Error has been used.

Model was trained for 250 epochs using batches of size 50 and produced an accuracy averaging around 64% and pictures looking close to realistic.

**Results and Discussion**

Once the model has been trained for \_ epochs until the training and validation losses converge and there is only little overfitting. The entire model and its weights are saved as “.h5” files so that they can be used to test/evaluate the model or can be loaded to train the model on a new set of data.

Once the .h5 file is loaded and when tested against the validation data results obtained are as follows:

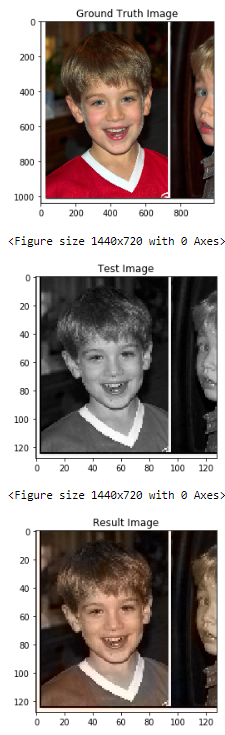
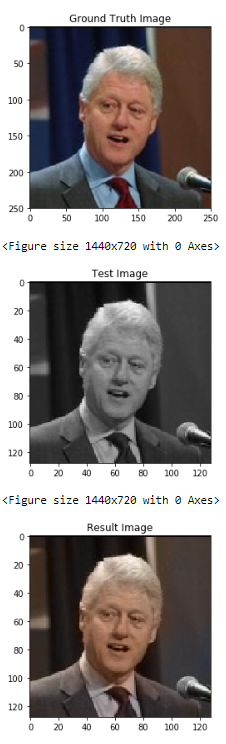
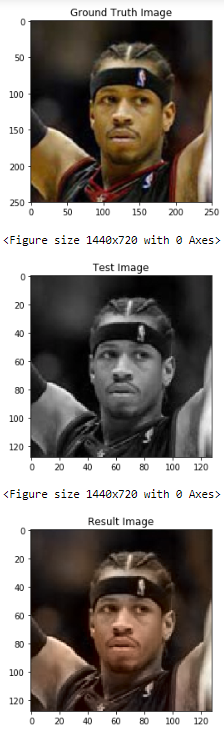
 

Figure Results on Validation Data

Next the model was evaluated on images it has never seen. To do this, images from the web were downloaded and passed to the model to see results.

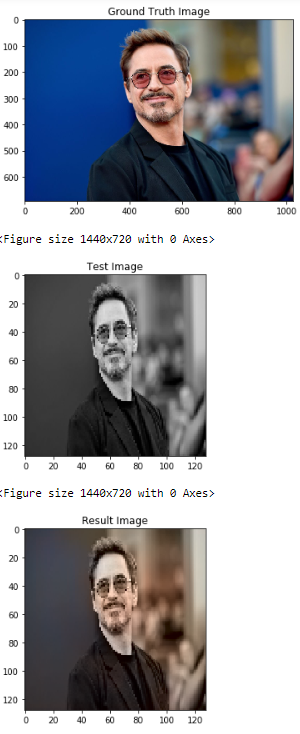
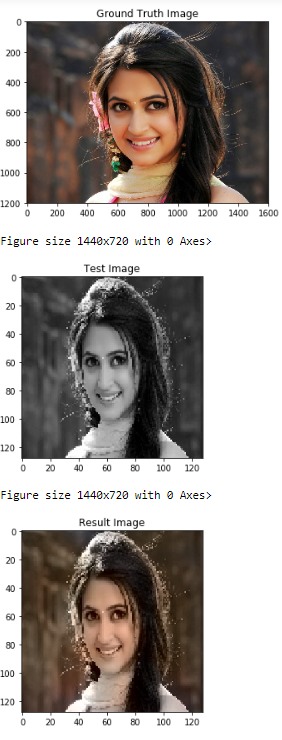
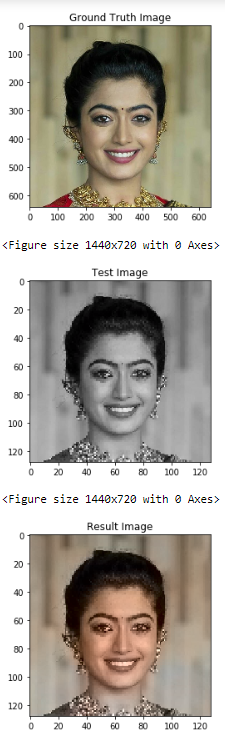


Figure Results obtained on Unseen Colored Images

The results although slightly under saturated looked convincing. When the result image is shown to a person, they look convincing enough to fool the person to think it is an original.

Next step in the evaluation of the model is to see how it performs with actual greyscale images and see the results produced.

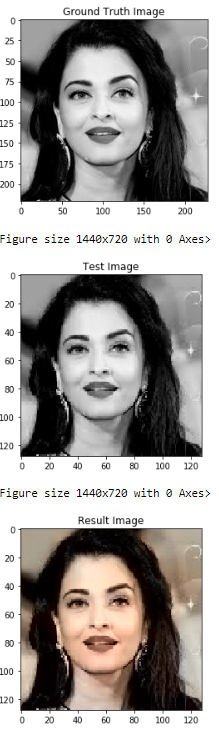
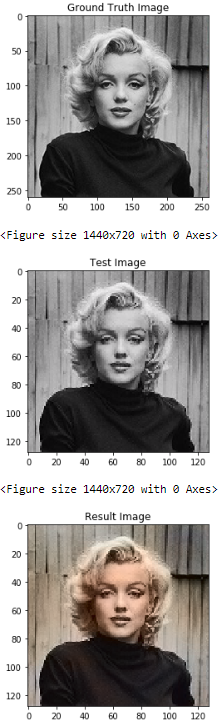
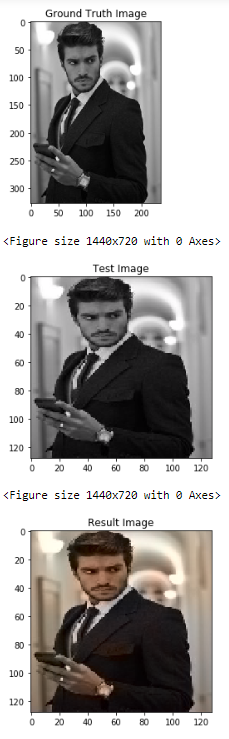


Figure Result on greyscale images

**Conclusion**

* It can be observed from the results obtained that the images tend to have a slight brownish hue to them.

Upon researching the cause for this phenomenon, it has come to know that brown color tends to produce minimum MSE when model minimizes loss during training by adjusting parameters. The parameters result in a and b values which give a brownish tone to the resulting image.

* Due to the limitations in time and compute power the model developed has been trained only on images with humans since the most common subject in greyscale images are humans.

Given enough resources the model can be trained on much diverse data allowing to perform much better in different scenarios.

* During the research for the project many implementations of colorization of greyscale images were encountered. These were very advanced utilizing Generative Adversarial Networks, Transfer learning techniques using Residual Networks, etc. These implementations perform exponentially better than the project implemented but developing such models require large resources and data which seemed out of scope.

**References**

[1] Richard Zhang, Phillip Isola, Alexei A. Efros. *Colorful Image Colorization.*

University of California, Berkeley

[2] Zezhou Cheng, Qingxiong Yang, Bin Sheng. *Deep Colorization. 30 Apr 2016*

[3] Qiwen Fu, Wei-Ting Hsu, Mu-Heng Yang *Colorization Using ConvNet and GAN.*

Stanford University.

[4] Z. Cheng, Q. Yang, and B. Sheng. *Deep colorization*. In Proceedings of the IEEE International Conference on Computer Vision, pages 415–423, 2015.

[5] Kr¨ahenb¨uhl, P., Doersch, C., Donahue, J., Darrell, T.: *Datadependent initializations of convolutional neural networks.* International Conference on Learning Representations (2016)

[6] Mingming he , Dongdong chen , Jing liao , Pedro v. Sander , and Lu yuan.

*Deep Exemplar-based Colorization.*

University of Science and Technology of China, Microsoft Research.

[7] Dahl, R.: *Automatic colorization.* In: http://tinyclouds.org/colorize/. (2016)

[8] Ya Le, Xuan Yang. *Tiny ImageNet Visual Recognition Challenge.*

<https://tiny-imagenet.herokuapp.com/>