“Image Colorization with Deep Convolutional Neural Networks”

A Project

Presented to the

University of Dhaka, Bangladesh

In Partial Fulfillment

Of the Requirements for the Degree

Master in Information Technology (MIT)

By

Tahseen Zaman

132032

2019-20

SIGNATURE PAGE

PROJECT	: “Image Colorization with Deep Convolutional Neural Networks”
AUTHOR	: Tahseen Zaman
DATE SUBMITTED	: 17th November 2020
SUPERVISED BY	: Dr. Ahmedul Kabir Assistant Professor Institute of Information Technology, University of Dhaka
SUPERVISOR’S	<Signature and Date>
APPROVAL	: ______________________________________

ACKNOWLEDGEMENTS

This project would not be possible without the inspiration and help from many people. I will express my gratitude towards my dear friend, mentor, and excellent cheerleader Zelato my sister first.

And of course without the guidance from my supervisor Dr. Ahmedul Kabir this project wouldn’t be as outstanding. His detailed instruction, careful selection of materials, and above all his hopeful nature about this project pushed me further to make finer and finer tweaks to make the implementations praiseworthy. I wish him the very best.

ABSTRACT

One Convolutional Neural network-based encoder-decoder model faithfully colorizes black and white images without any human assistance while able to preserve the natural tonality. We gave best efforts to produce more aesthetically pleasing output than many existing solutions based on regression or classification. Moreover, it's much more time-efficient than the traditional method to colorize by human assistance. Without the need to preprocess the data for feature extraction the model can work in conjunction with a VGG16, a domain adaptation model to learn image features, widely in use for computer vision applications. We have explored many cutting-edge concepts regarding color spaces, Convolutional Neural Network architecture, Encoder-Decoder type domain adaptation, Transfer learning, regularization, model evaluation, etc.

TABLE OF CONTENTS

LIST OF TABLES

TABLE OF FIGURES

INTRODUCTION

One of the most sought-after methodologies is colorizing medical, mining, or general aesthetic pictures. Most of the time these pictures are represented with Black and white with different contrasts but it does not convey much information about the tone and hue of the shapes. To better understand and identify different parts, it is more useful to use colorized images instead of B&W. This project deals with this necessity. There are many pre-existing methodologies to colorize images; several require human assistance but our deep learning-based model can produce colorized images without human assistance by identifying the image features while using Statistical Methods [1] [2] [3].

Our Convolutional Neural Network (CNN) accepts black and white images as input and generates a colorized version of the images as its output, Figure 1: Input and output; shows an example of such a pair of input and output. Here black and white image was feed into our model, and a colored version is produced purely based on Deep Convolutional Neural Network architecture [2].

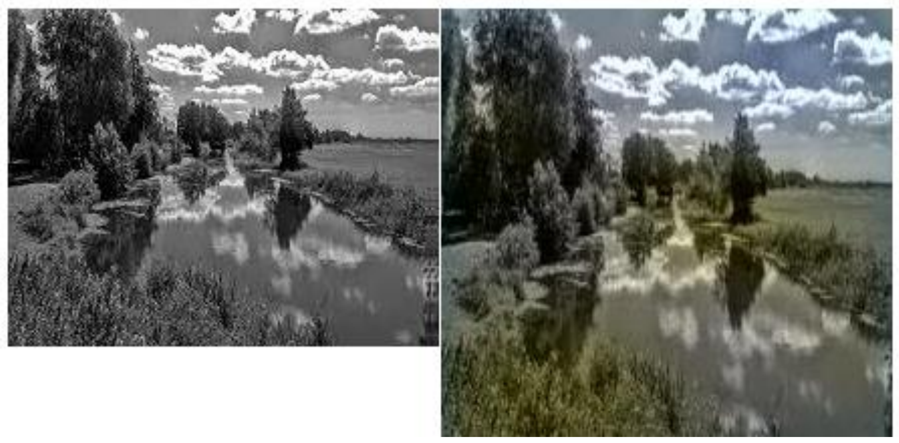
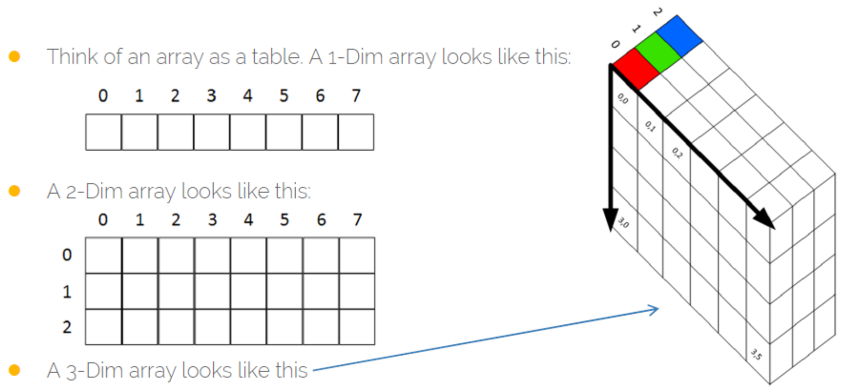


Figure 1: Input and output

METHODOLOGY

Generally, the computer stores image data as either a 2-D array in the case of Black and white images or as a 3-D array in the case of color images. Then 3-D array each 2-D array represents one of the color channels of the image; as shown in Figure2: Image Data. The information stored



Think of an array as a table. A 1-Dim array looks like this:
A 2-Dim array looks like this:
A 3-Dim array looks like this

Figure 2: Image Data

in channels of images varies from color space to color spaces. For our calculations, we are using LAB color space. There is a popular RGB, CMYK, or HSV, illustrated in Figure 3: Color matrix and LAB is shown in Figure 4: LAB Color space [4] [5].

LAB color space one channel is allocated for Black and white only. Then one for Blue to yellow and another channel for green to red in the color wheel. This simplifies our problem, we are predicting the value of two colored channels for one given value for the black and white channel. If it’s with other channels we then have to process three channels instead of two [6].

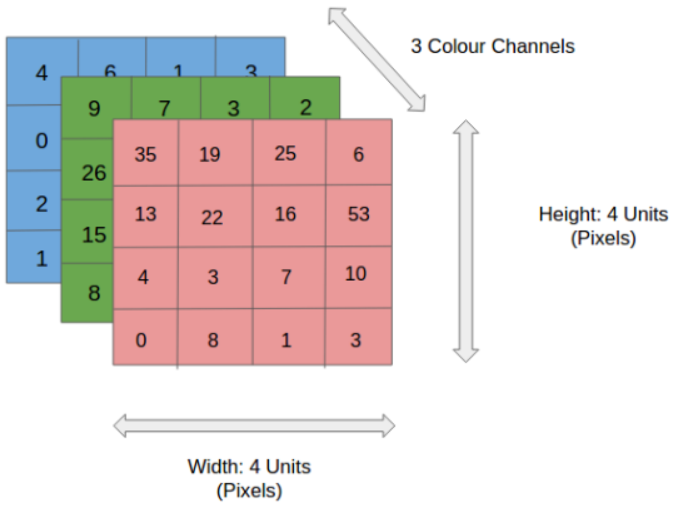


Figure 3: Color Matrix

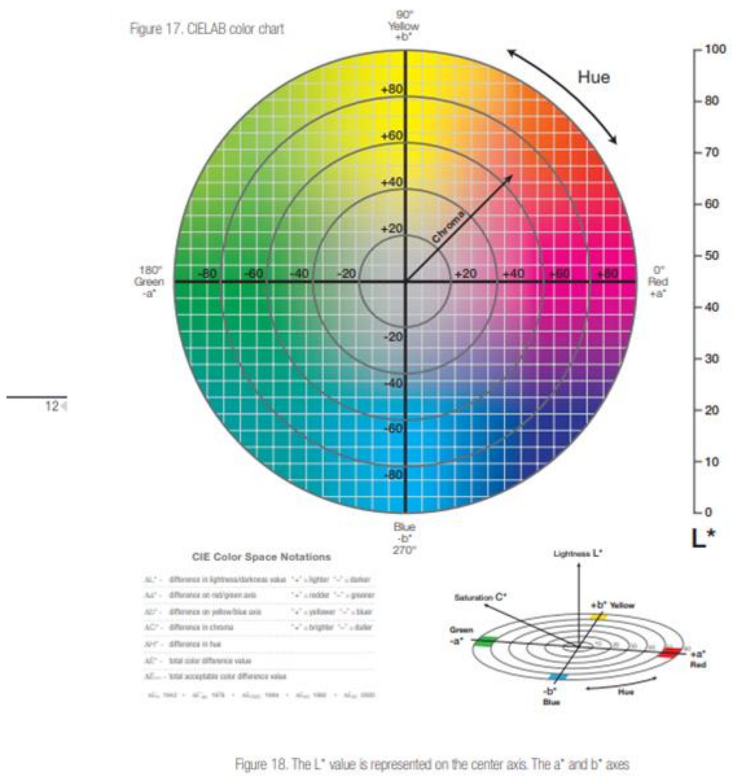


Figure 4: LAB Color Space

Our Deep learning model is a Convolutional Neural Network-based, Encoder, and Decoder, an example of such architecture is Figure 5: Encoder and Decoder Model. By using CNN based neural

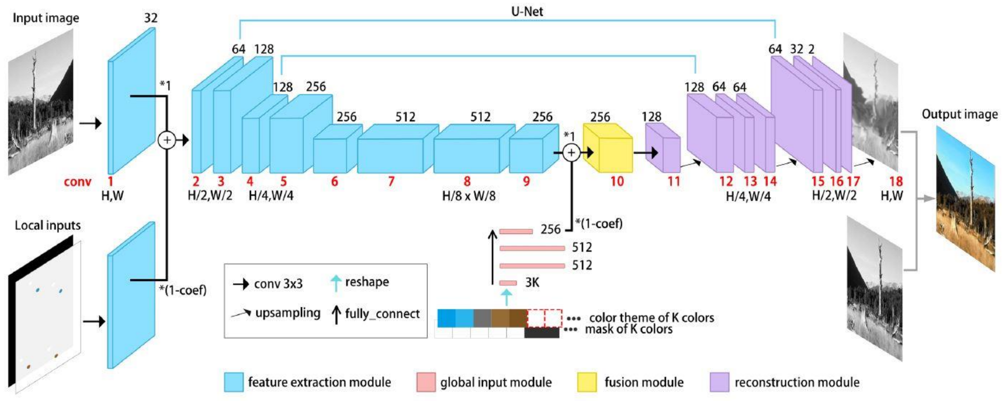


Figure 5: Encoder and Decoder Model

network we ensure- much less computational power will be required and the spatial data for any relative positional pixel data will not be lost. We are not required to preprocess the data for and features [2] [4] [5]. The model will learn different features of images in the training process. By using existing VGG-16 we have saved computational burden for the encoder part, integrating the first 18 layers with an autoencoder system with residual connections that merge intermediate outputs, the Encoder's output with those produced by the latter decoding portion of the network [7]. This is a form of Transfer learning used mostly in a different application of domain adaptation architecture.

IMPLEMENTATION

The whole implementation is in Python. We used Keras for implementing our Encoder-decoder architecture. For preprocessing the images OpenCV was used. This implementation is heavily dependent on Matrix manipulation which was done using NumPy.

Availability of high quality and complete data is required for any Machine learning model to build upon. The accuracy and coverage depend mostly on training data. We used Google's Nature and landscape data and Outdoor and static dataset and then Google's dataset of humans for ensuring verity while training our model, MIT CVCL Urban and Natural Scene for better representing the natural landscape [4] [2].

1: Table of Dataset

Dataset name	Size	Categorical
Landscape and Nature	2,688	Landscape and Nature
Human Gesture recognition	4,000	Human
MIT CVCL Urban and Natural Scene	4,319	Landscape and Nature
Wallpapers from web	15,000	Landscape and Nature

This is to be noted while these datasets are labeled via subject our model doesn't require the training set to be in categories or the labeling.

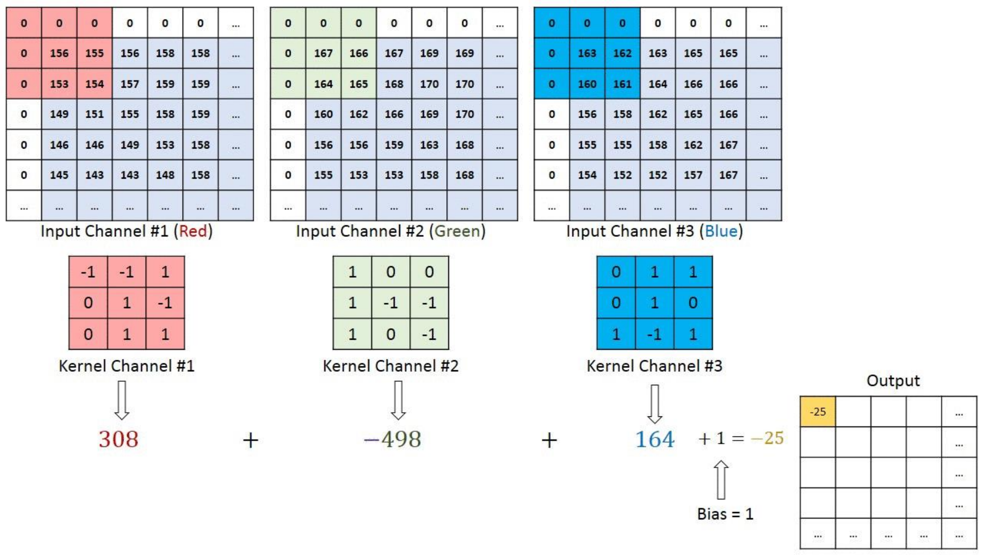


Figure 4: Convolution Matrix

We preprocess all images to the LAB color space. For each of the images to we extract the training label data in this case channel of Black and white and the target column is the image's two-colored channels.

We load the images as 224x224x3 the same as the input size of VGG16. As all Data in LAB color space is in the range of -127 to 127, we normalize the pixel values by dividing them by 128 [6] [4] [5].

The encoder part of our model is the Convolutional layer followed by the pooling layer. This model is of depth 5, that's five layers of convolution as shown in Figure 6: Kernel Convolution and pooling for the encoder side, here the matrices are downsampled. At the decoder side of our model, the downsampled 3D matrix is then upsampled and de-convolution happens in this phase. This sequential model is shown in Figure 7: Convolutional layer, followed by pooling. The Decoder is shown in Figure 8: Decoder Architecture.

The model generates two arrays of each of dimension 224x224x1, corresponding to the colored channel. The three channels are then concatenated together to form a LAB representation of the predicted image.

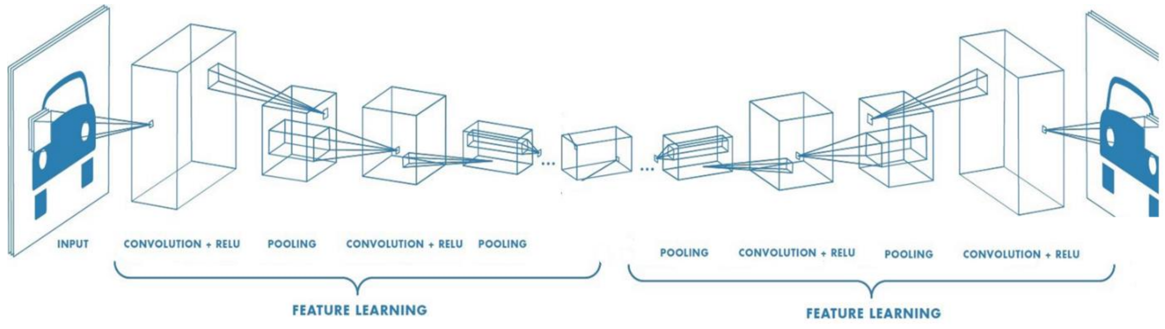
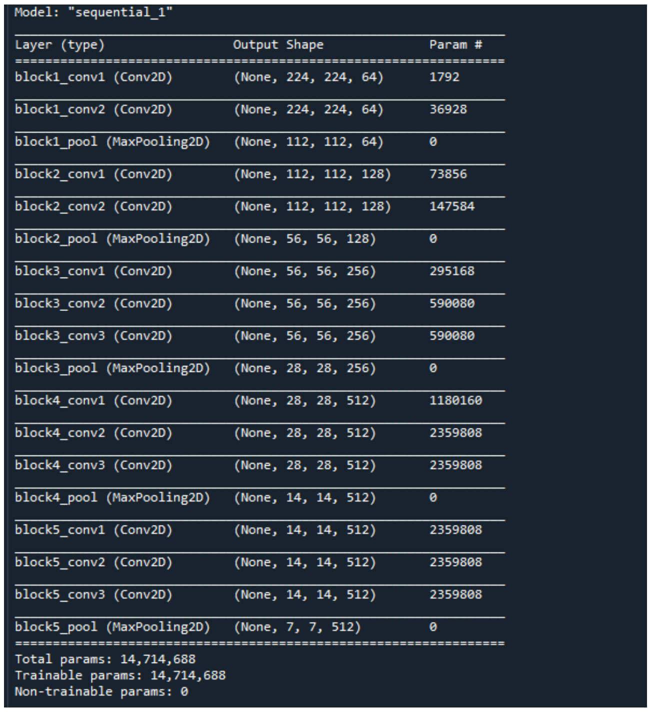


Figure 5: Architecture of Convolution and pooling layer

L1 regularization (Lasso Regression) and L2 regularization (Ridge Regression) helped with the primary overfitting, shown in Figure 9: Overfitting, of data [7] [8]. This is further broken down in the Result section. MSE (Mean Square Error) loss function was used, the model was trained for in total of 1500 epochs for more than 24,000 image data. The testing accuracy was 83% and Validation accuracy was 76%.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

Figure 6: Decoder Model Summary

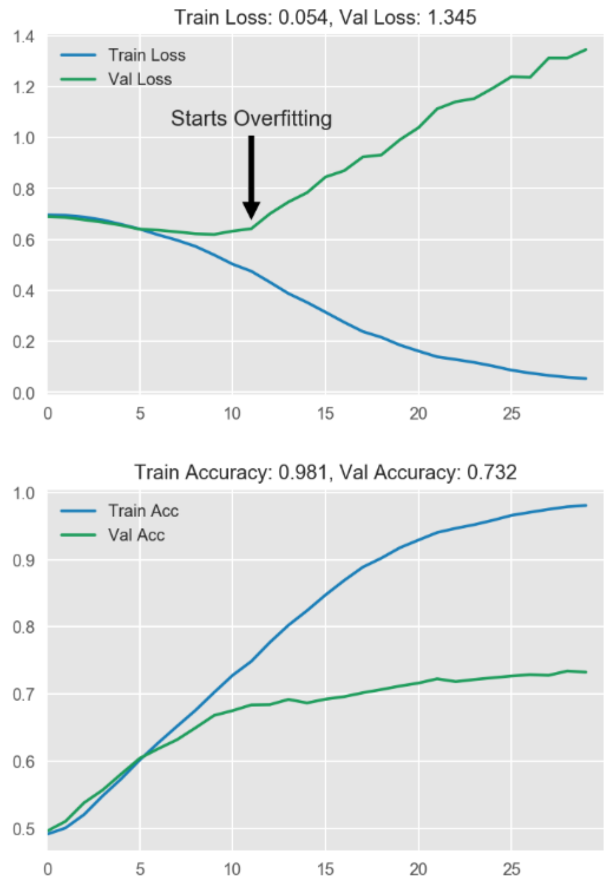


Figure 7: Over Fitting

RESULTS

Without Regularization and with regularization comparisons:

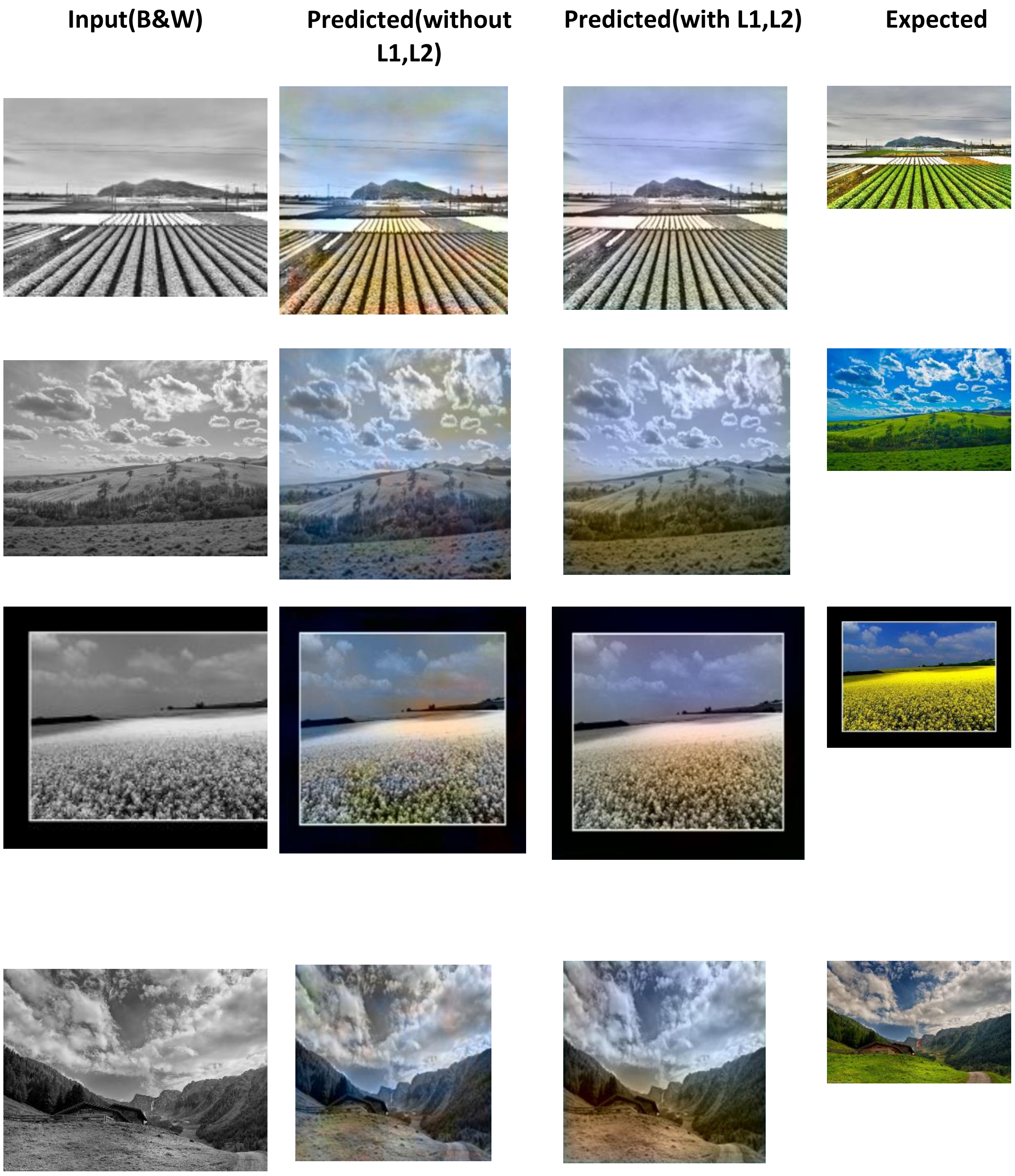


Figure 8: Comparison between Models

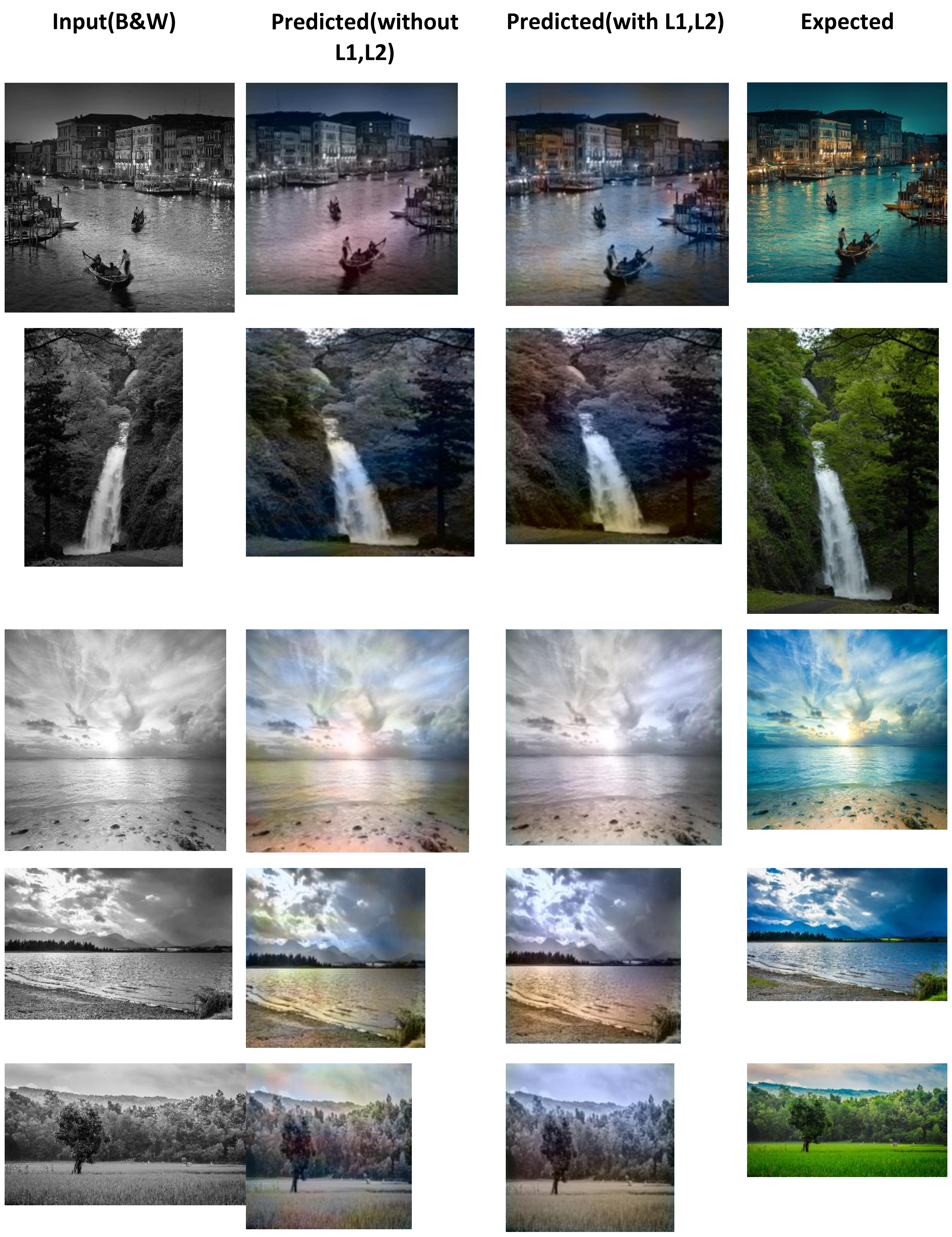


Figure 9: Comparison between Models

Our model performs remarkably well while coloring foliage, skies, and especially grass. We notice in many cases the predicted photos are sepia-toned and muted-toned. In the case of multicolored objects or photos that should produce the same hue with different brightness, our model seems to struggle and produces the same unattractive, subdued mixture of possible colors, model performs poorly in these cases. These characteristics are shown mostly because of overfitting. By introducing regularization we were able to overcome some of the muteness and able to produce more vibrant colors. This also increased the accuracy of the model overall.

In the case of human skin and clothes, our model struggles to produce punchier color as it tends to generalize the scenarios [2]. As human skin falls under a very wide range of colors from fair to

Input Output

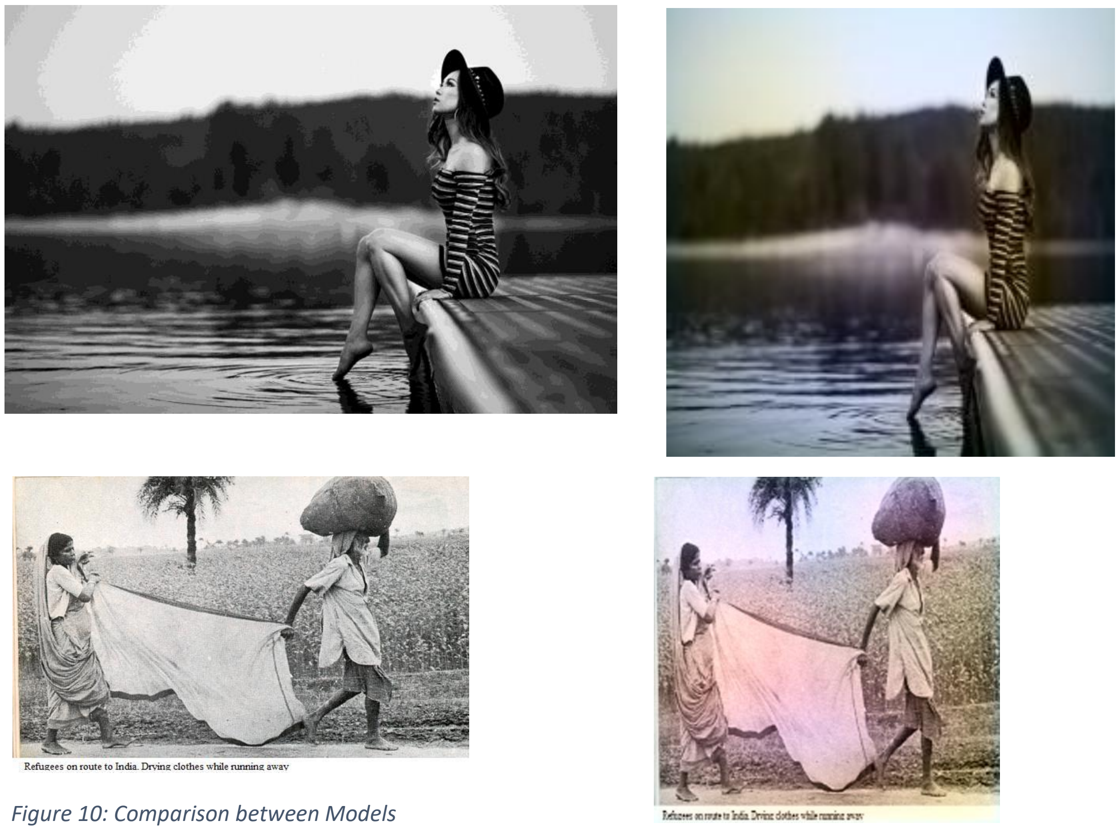

Refugees on route to India. Drying clothes while running away
Refugees on route to India. Drying clothes while running away

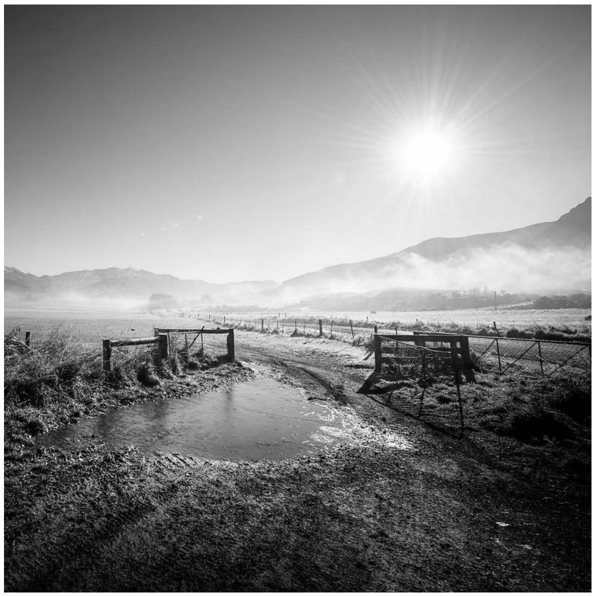
Figure 10: Comparison between Models

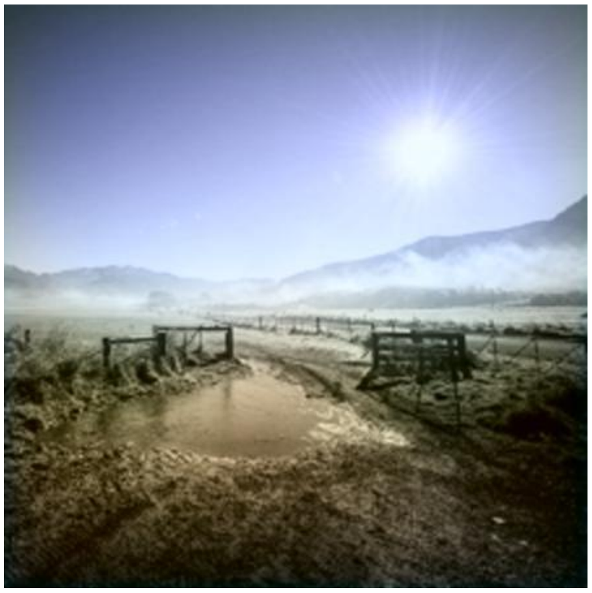
dark brown, it's understandable a trained model without human assistance will struggle to produce the appropriate color of skin. In most cases, the skin is colored Sepia. This is illustrated in the figures.

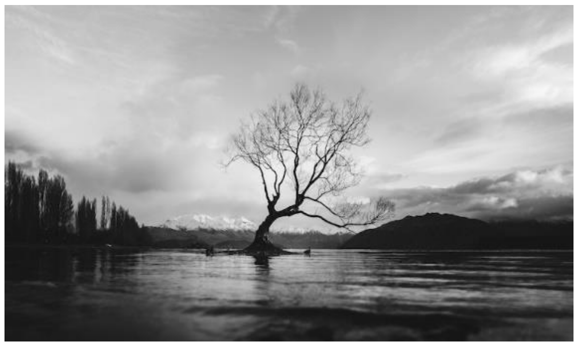
In the case of still printing it may be the case that virtually none of the colors match the color responding real-life scene. Nonetheless, the output is still aesthetically pleasing. However, there exists a noticeable amount of noise in the classification results, with blobs of various colors interspersed throughout the image [2]. This may result from several factors. It may be the case that the system discretizes the A and B channels into bins that are too large, which means that image regions that contain color gradients may appear choppier [1].

More examples on Grass and Sky:

Input **Output**







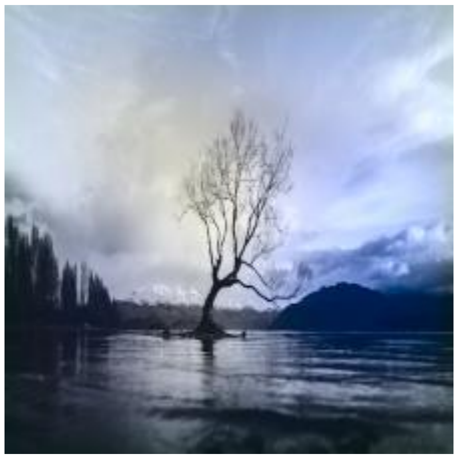


Figure 11: Comparison between Models

CONCLUSION

This model works wonders for simple pictures with no complexity of a subject, like mountains, skies. In the future we should look into a different method of Fusion with CNN architecture, two models can separately be trained one for better feature extraction and another for colorization. As we are unable to add punchier, vibrant color the model can be checked for other methods of Encoder-Decoder implementation other than CNN type. We can also utilize post-processing schemes such as total variation minimization and conditional random fields to achieve a better dynamic range.

REFERENCES

[1] P. A. R. G. a. J. M. B. Hariharan, "Hyper columns for object segmentation and fine-grained localization".

[2] X. G. a. Y. Bengio, Understanding the difficulty of training deep feedforward neural networks. In International conference on artificial intelligence and statistics.

[3] X. Z. S. R. a. J. S. K. He, "Deep residual learning for image recognition".

[4] [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2.

[5] [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798.

[6] P. I. A. A. E. Richard Zhang, "Colorful Image Colorization".

[7] J. H. a. Y. Zhou, "Image Colorization with Deep Convolutional Neural Networks," IEEE.

[8] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks".

APPENDIX A

Machine learning workflow:

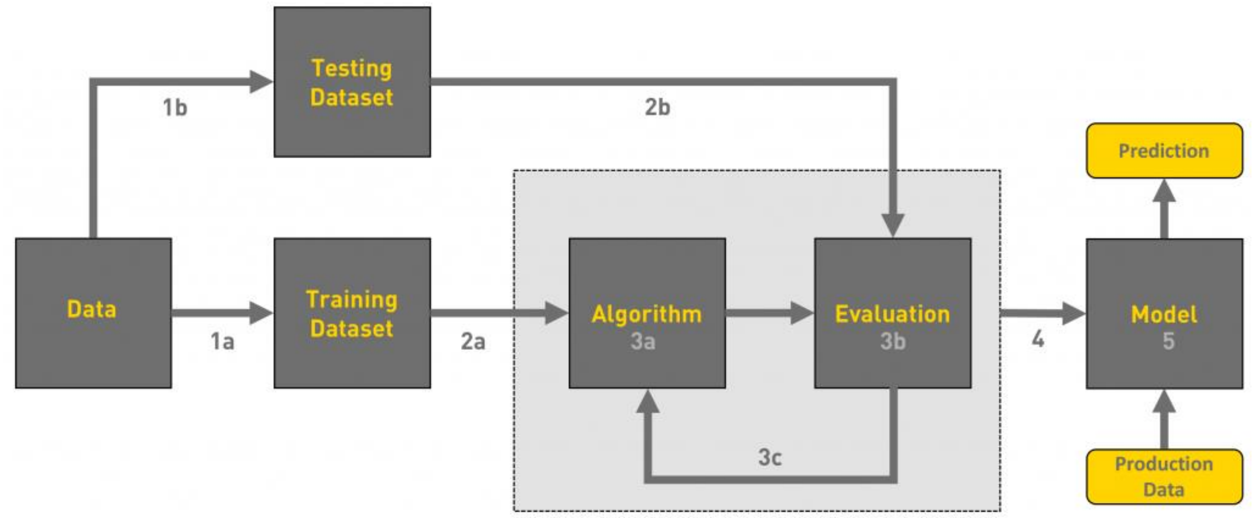


Figure 12: Machine Learning Work Flow

Machine Learning is the study of making machines more human-like in their behavior and decisions by giving them the ability to learn and develop their own models. This is done with minimum human intervention, i.e., no explicit programming. The learning process is automated and improved based on the experiences of the machines throughout the process. Good quality data is fed to the machines, and different algorithms are used to build ML models to train the machines on this data. The choice of algorithm depends on the type of data at hand, and the type of activity that needs to be automated. The general steps of Machine learning are as follows:

1. Gathering data
2. Data pre-processing
3. Researching the model that will be best for the type of data
4. Training and testing the model
5. Evaluation