COLORIZATION OF GRAYSCALE IMAGES

USING GENERATIVE ADVERSARIAL NETWORK

A Thesis submitted in partial fulfillment of the requirement for the Award of the Degree of

MASTER OF ENGINEERING

in

Electronics and Communication

Submitted By

UMA SHARMA

801761018

Under Supervision of

Dr. Vinay Kumar

Associate Professor, ECED

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

JULY, 2019

DECLARATION

I, **Uma Sharma** hereby declare that the work presented in this thesis entitled "**Colorization Of Grayscale Images Using Generative Adversarial Network**" in partial fulfillment of the requirement for the award of degree of **Master of Engineering (ECE)** submitted at **Electronics And Communication Engineering Department**, Thapar Institute of Engineering & Technology (Deemed to be University), Patiala is an authentic record of work carried out under supervision of **Dr. Vinay Kumar(Assistant Professor, Electronics And Communication Engineering Department)**, Thapar Institute of Engineering & Technology (Deemed to be University). The matter presented in this has not been submitted either in part or full to any other university or institute for the award of any other degree.

Uma Sharma

(Uma Sharma)

Date: 6/August/2019

(801761018)

It is certified that the above statement made by the candidate is correct to the best of my knowledge and belief.

Dr. Vinay Kumar

Date: 6/August/2019

Assistant Professor

Department of Electronics And Communication Engineering

Thapar Institute of Engineering & Technology

(A deemed tobe University), Patiala, Punjab

ii

ACKNOWLEDGEMENT

I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of this dissertation. I acknowledge with gratitude and humility my indebtedness to **Dr. Vinay Kumar**, **Associate Professor**, ECED, Thapar Institute of Engineering and Technology (Deemed to be University), Patiala under whose guidance I had the privilege to complete this dissertation. I wish to express my deep gratitude towards him for providing individual guidance and support throughout the dissertation work.

I convey my sincere thanks to **Head of the Department**, **Dr. Alpana Agarwal**, entire faculty and staff of Electronics and Communication Engineering Department for their encouragement and cooperation. .

I am highly obliged and wish to owe my sincere gratitude to **Dr. Amit Mishra**, **Assistant Professor**, ECE Program Coordinator and **Dr. Hem Dutt Joshi**, **Assistant Professor**, P.G. Coordinator, Electronics and Communication Engineering Department, Thapar Institute of Engineering and Technology (Deemed to be University), Patiala for providing me facilities, learning atmosphere and infrastructure in ECED. My greatest thanks are to all who wished me success especially my family. Above all I render my gratitude to the Almighty who bestowed ability and strength in me to complete this work.

Uma Sharma

ABSTRACT

The automatic colorization of the grayscale images is an appealing area in the field of image processing**.** It has a major application in the restoring of aged or degraded images. In this work, we perform the colorization of grayscale images using a Generative Adversarial Network. This method requires less human interference. Especially, a conditional generative network is used to get the desired output. A piece of extra information or condition is fed to both generator and discriminator model. In this thesis, two color spaces have been used: CIE-LAB color space and YCbCr color space. Model is trained over both the color spaces and their results are compared. Experimental results from both the color spaces show that the CIE-LAB color space is more accurate than the YCbCr color space. Images generated by model trained over CIE-LAB color space are sharper and brighter.

TABLE OF CONTENTS

Sr No	Name of the Chapters	Page No.
Declaration		***ii***
Acknowledgement		***iii***
Abstract		***iv***
Table of Contents		***v-vi***
List of Figures		***vii***
List of Glossary		***viii***
Chapter 1	**INTODUCTION AND STATEMENT ANALYSIS**	
1.1 Intoduction To Image Processing		1
1.2 Types Of Images		2
1.3 Convolutional Neural Network		3
1.3.1 Basic concepts		3
1.3.1.1 Layer		5
1.3.1.2 Convolutional layer		5
1.3.1.3 Pooling layer		7
1.3.1.4 Training		7
1.3.1.5 Loss function		8
1.3.1.6 Dropout		9
1.3.1.7 Normalization		9
1.4 Color Space		10
1.5 Problem Statement		13
Chapter 2	**GENERATIVE ADVERSARIAL NETWORK**	14
2.1 Introduction		14
2.2 Generative vs. Discriminative Algorithms		15

2.3 Training Of A GAN 16

2.4 Types Of GANs 16

Chapter 3 LITERATURE REVIEW 19

3.1 Colorization Of Grayscale Images 19

3.2 Colorization In Medical Field 23

Chapter 4 TRAINING OF THE GAN 26

4.1 Introduction 26

4.1.1 Importing packages 26

4.1.2 Reading images from the dataset 27

4.1.3 Training and testing set 27

4.1.4 RGB to LAB/YCbCr 27

4.1.5 Generator model 28

4.1.6 Discriminator model 32

4.1.7 GAN 32

4.1.8 Training 33

Chapter 5 RESULT AND DISCUSSION 35

5.1 Results Using LAB Color Space 35

5.2 Results Using YCbCr Color Space 36

5.3 Comparison Of Results From Both Color Spaces 37

5.4 Model Graphs 38

Chapter 6 CONCLUSION AND FUTURE SCOPE 40

6.1 Conclusion 40

6.2 Future work 40

REFERENCES 41

LIST OF FIGURES

Sr. No	Figure Details	Page No
Figure 1.1	Image	1
Figure 1.2	Convolutional neural network	4
Figure 1.3	Image matrix convolving with the filter matrix	6
Figure 1.4	Resultant matrix after convolution	6
Figure 1.5	RGB color model	11
Figure 1.6	LAB color model	11
Figure 1.7	YCbCr color model	12
Figure 2.1	A GAN network	14
Figure 2.2	Conditional Generative Adversarial Network	17
Figure 4.1	Generator model	31
Figure 5.1	Results with LAB color space (a) Original image (b) Image generated using GAN	35
Figure 5.2	Results with YCbCr color space (a) Original image (b) Image generated using GAN	36
Figure 5.3	Comparison (a) Real image (b) GAN output using CIELAB color space (c) GAN output using YCbCr color space	37
Figure 5.4	Graph between gan accuracy and *gen1_model* loss	38
Figure 5.5	Graph between gan accuracy and *gen2_model* loss	39

LIST OF GLOSSARY

GAN	Generative Adversarial Network
CNN	Convolutional Neural Network
ICC	International Color Consortium
CIE	International Commission on Illumination
C-GAN	Conditional Generative Adversarial Network
DC-GAN	Deep Convolutional Generative Adversarial Network
LAP-GAN	Laplacian Pyramid Generative Adversarial Network
SURF	Speeded Up Robust Features
SR-GAN	Super Resolution Generative Adversarial Network
BicoGAN	Bidirectional Conditional Generative Adversarial Network
BiGAN	Bidirectional Generative Adversarial Network
MedGAN	Medical Generative Adversarial Network
PET	Positron Emission Tomography
CT	Computed Tomography
NIR	Near Infrared

CHAPTER 1

INTRODUCTION AND STATEMENT ANALYSIS

1.1 INTRODUCTION TO IMAGE PROCESSING

Image processing is a form of signal processing in which the signal is an image. We perform different operations on an input image to get an enhanced version of it or to get some useful information from it. Today, image processing is among fastly growing technologies. Image processing primarily consists of three steps:

- Getting an input image from some sources
- Performing various operations on that image
- Output image

Figure 1.1 Image [1]

Visual perception of anything is defined as an image. Figure 1.1 shows an image of a bird. Everything what we see or what we analyze about something can be represented in the form of an image. An Image can be two dimensional or three dimensional. A two dimensional image can be described as A(u,v), where u and v represent the spatial coordinates. For a pair of coordinates (u,v) the signal level of A defines the intensity of the image at that particular point.

1.2 TYPES OF IMAGES:

1. Binary image- It contains only two-pixel elements 0 and 1, where 0 represents the black and 1 to white.

2. Black and white image- This image consist of only black and white color.

3. 8 Bit color format- This represents a grayscale image. It has 256 different shades, where 0 defines the black color, 255 defines the white color and 127 stands for graycolor.

4. 16 Bit color format- It has a total of 65,536 values. This 16-bit color format is subdivided into RGB format.

More information can be extracted out from a color image rather than a grayscale image. So in this thesis, we will deal with the colorization of the grayscale images using the generative adversarial network.

In the field of image processing, the automatic colorization of the grayscale images is an appealing area. Adding colors to the images can improve the amount of information contained in that image. In biomedical imaging, most of the medical equipment reproduces their output in grayscale format. A human eye is incapable of distinguishing more than a few hues of gray, so here the colorization process is very helpful [2].

Three values per pixel are represented by a color image whereas a grayscale image only represents the brightness level of the image. It has values ranges from 0 (black) to 255 (white). So for the computer, a 10x10 pixel grayscale matrix is simply a list of 100 values where each value represents the brightness level of a certain pixel. Transforming a colored image into a grayscale image is a straightforward task because we have to drop off some information relevant to the color of the image. But when we want to perform the reverse of it, means if we want to transform a grayscale image into a colored one then the process becomes tough. The problem is that within a single grey level there are a number of colors get fitted up means there is many to one mapping. It is hardly known which color corresponds to that particular grey level. So synthesizing a colored image into a grayscale image is not an easy task.

In early times this colorizing process was done manually in photoshops which is very time-consuming. In manual coloring, color for each pixel has to be decided manually which is a tiresome and time-consuming process.

So in this thesis, we have used an automatic method for colorization of the grayscale images. Here we are using generative adversarial network for this purpose. Generative adversarial networks (GANs) are a powerful class of deep neural networks [3]. Various image to image translation operations is performed using this GAN architecture. Colorization of a grayscale image is one of them. GAN provides us a way to automatically colorize the pictures by getting trained over thousands of images. The architecture examines the pattern in the images and based on that pattern the colorization is performed.

1.3 INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORK

Convolutional neural networks are made up of neurons. These networks are capable of understanding the various patterns in an image through a learning process. GANs are made up of CNNs. Further, these GANs are used for pattern recognizations [4], various image to image translation [5], image classification [6], anomaly detection and mitosis detection.

When compared to the other architectures of Artificial Neural Network CNN are different from them because instead of focusing on all over the problem they only deal with a specific type of input feature [4]. This makes the architecture of CNN simpler as compared to others.

So in this chapter, we are going to discuss the convolutional neural network and its different layers. A neural network in machine learning corresponds to a method to copy the same neurological processes that occur in a human brain. Neurons, like those in our brains, are the basic cells of convolutional neural networks. Convolutional neural networks are a category of neural network that consists of convolutional layers. This network can easily operate on two dimensional or 3-dimensional images. So they are the most common network used for image relevant tasks. CNNs are a standardized approximation of multilayer perceptrons which refers to the fully connected layers means each neuron in one layer is connected to all other neurons in the upcoming layer. The connectivity pattern of the neurons in the CNN are inspired by the biological processes.

1.3.1 Basic concepts

In this section, we will discuss the architecture of CNN's. A number of layers are there within a convolutional neural network called hidden layers.

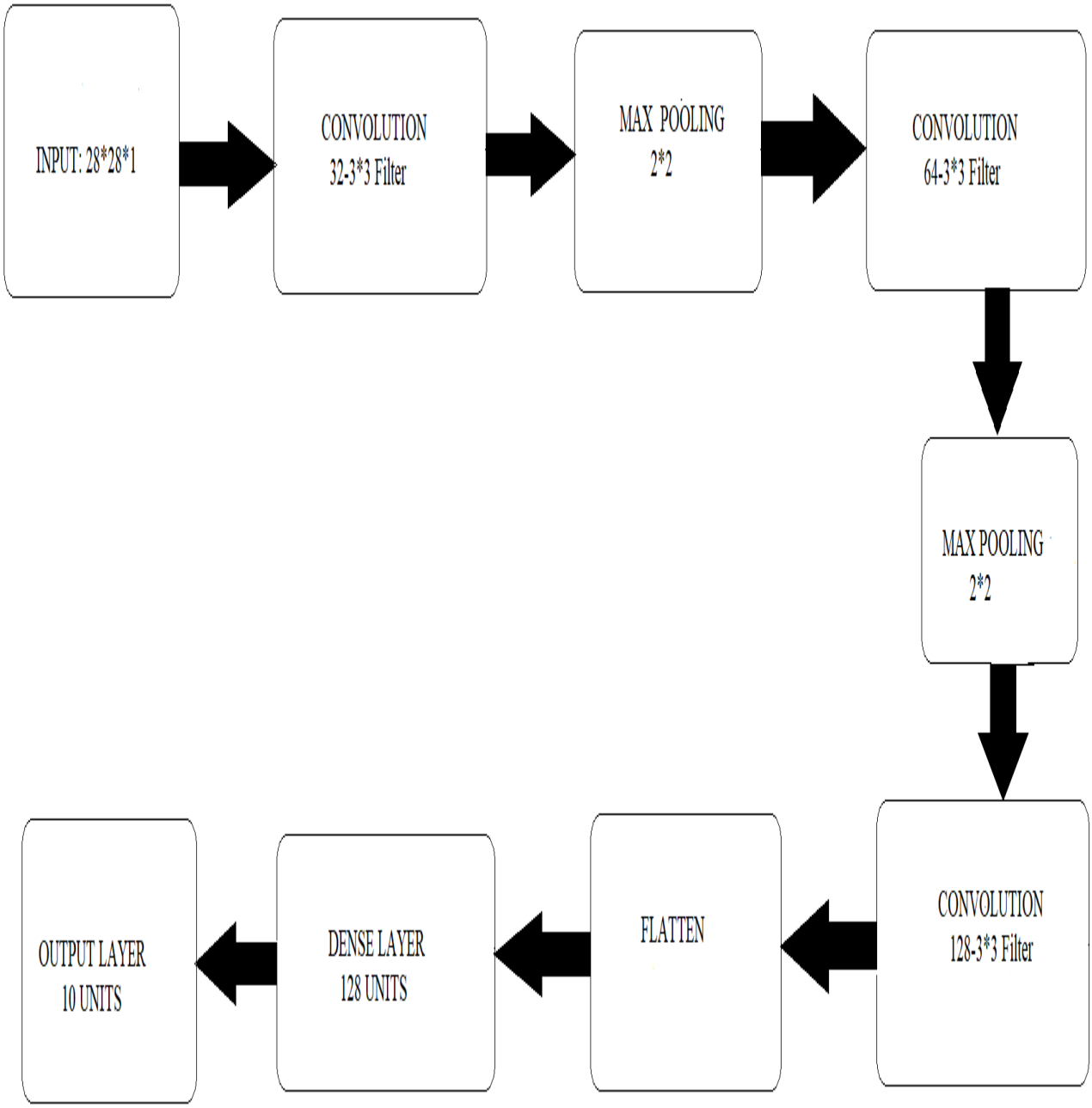


Figure 1.2 Convolutional neural network [7]

Here, Figure 1.2 shows different layers inside a convolutional neural network. As the input image passes through all layers one by one, at each layer the shape and parameters of the image get updated.

1.3.1.1 Layer:

The basic building blocks of CNN is a layer. A layer takes an input image, the image can be represented in the form of an array and after performing some operation an output image is produced. The output function can be realized as :

$$y=\sigma(Wa+b)$$

where a is input and y is output. W is the weight associated with that layer. σ is a nonlinear activation function and b is the bias vector. A layer possibly can take some inputs images and produce some output images. Further, these outputs are concatenated to produce a single output.

Weight matrix and bias vector are the parameters of the layer which learns through the process of backpropagation. These two parameters only provide a linear transformation. Using activation function non-linear mapping in the network is learned. Most common activation functions include the sigmoid function:

$$\sigma(x)=1/(1+e^{-x})$$

$$\sigma(x)=\tanh(x)$$

a hyperbolic tangent. The role of all activation function is to introduce non-linearity in the network. Activation function can be represented as a separate layer in CNN with identity matrix W and setting vector b as 0.

1.3.1.2 Convolutional layer:

The convolutional layer is the first layer in CNN. The function of a layer is to extract the various features from the input image. By using small blocks of input data the features from images are learned and the relationship between the pixels is preserved. Convolution is a mathematical operation that takes two inputs one is as an image matrix and second is a filter that also called the kernel. Take a 5x5 input image whose pixel values are 0 and 1. A filter matrix of 3x3.

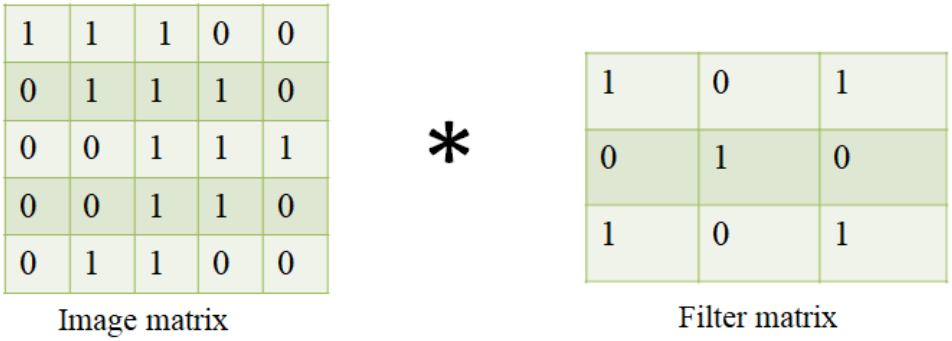


Figure 1.3 Image matrix convolving with the filter matrix

4	3	4
2	4	3
2	3	4

Figure 1.4 Resultant matrix after convolution

Figure 1.3 shows convolution of an image matrix of size 5x5 with a filter matrix of size 3x3. During convolution, its main hyperparameter is the size of its kernel. The size of the kernel can be non-square but it is not applicative for this thesis to consider such cases. Figure 1.4 shows the resultant matrix. Different filter matrix can be used to perform the various functions in an image like the blurring of the image, edge detection, and image sharpening.

Before the output is concerned, there are various parameters traditionally associated with a convolution layer. A layer does not consist of a single kernel, there are several kernels learned instead, and their outputs are spatially stacked over each other to get the final output. If an image of size 256x256 is given as input to a convolutional layer having 64 filters then its output will be of shape 256x256x64. For every point of input, it is not necessary a convolution calculation to be done there. It means we can skip every nth input point in either direction. This concept is called **stride.** Stride is the number of pixels that shifts over the input matrix at a time**.** It essentially allows us to downsample input resolution through convolution quickly. Upsampling can be implemented by transposed convolution, where zero padding is done to spread out the input

matrix. Upsampling provides output resolution higher than the input resolution. At the edges of the input image there are no values to calculate with, so convolution cannot be performed there. To avoid this problem padding is done in the input matrix, such that we can calculate the output even for edge points.

The convolutional layers can be stacked over another. A network in which several convolutional layers are stacked one over another is called a deep convolutional neural network or a deep CNN.

1.3.1.3 Pooling layer:

When images are too large, the pooling layer reduces the number of parameters of the images but retains the important information about the input image. It reduces the computations required during the processing. The information which is extracted from the input is rotational and positional invariant, which maintains the training process. The main goal of any pooling layer in a network is to achieve spatial invariance. Pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling takes the largest element from the area covered by the kernel. Average pooling gives an average of all values of the portion covered by the kernel. Sum pooling gives the sum of all values in the area covered by the kernel. Max pooling performs the de-noising along with dimensionality reduction. It discards the noisy activation. A max-pooling achieves the powerful results in capturing the invariance in images [8]. After going through all the above processes, the model is now enabled to learn the features of the input image.

1.3.1.4 Training:

Parameters of all the layers in the given network are combined which are collectively known as the network's weights. With the user-provided data the weights are learned and the method with which these weights are acquired is called training. The adjustment of weights or filter values within a network is done through backpropagation [9]. Initially, some random values are given to the network as weights. Maybe these values will fit in the proposed network or not. So the model is taught to change the values to minimize the error. This is done by training the model. Back-

propagation is one of the ways to train our model. Back propagation can be divided into four steps: the forward pass, the loss function, the backward pass, and the weight's update.

The first step is the forward pass, an input image is passed through the network's layers to get the prediction. Based on those predictions, the difference between the predicted value and ground truth value is find out. This value is known as a **loss** of the network. To minimize this loss the weights of all the layers should be updated. Here the concept of backward pass comes into the picture. The weights which contribute most to the loss are considered and are adjusted in such a manner to reduce the loss. The discrete partial derivative is calculated between the network's weights the loss function output as dLoss/dW, and in the initial weight, we update the weights as follows:

$$w=w_i - \eta \frac{dLoss}{dW}$$

η=learning rate

w_i=initial weight

The backward pass is applied to all the layers within the network in the opposite direction of the forward pass. The learning rate is chosen by the programmer. A higher learning rate results in big steps are taken in weight update, so the model converges to the optimal state in very little time. However, a learning rate that is too high could result in jumps that very large and not accurate enough to reach the optimal point. In a single training iteration forward pass, loss function, backward pass, and weight updates are done. For a fixed number of iterations, this process is repeated. A fixed set of training images called batch is taken. During the last training set, the network gets trained well enough so that the weights can be tuned to the correct values.

1.3.1.5 Loss function:

A network's loss or cost function is defined as a function of its prediction and the ground truth value. The value of this cost function measures the error of the prediction about the predefined label. It is always a positive value. Euclidean L2 norm and cross-entropy loss are the most commonly used functions. A loss function must conform to two assumptions in order to be applicable in any learning algorithm. First, the loss of the entire training set is decomposed into an average of loss functions of individual training examples:

$$F=\frac{1}{n}\sum_x F_x$$

To compute partial derivatives of the loss function for a single training example which is used in the back-propagation algorithm this assumption is necessary. This data-iterative optimization is called Stochastic Gradient Descent (SGD). Deep learning neural networks are trained by using these algorithms. The second assumption is that loss function must be a function output of the network. In a binary classification problem, cross-entropy is commonly used. For multi-classification multi-class cross-entropy is used. Cross-entropy determines the divergence between two probability distributions. If the cross entropy's value is large it means that the difference between two distribution is high, while if the cross entropy's value is small it means that two distributions are very much similar to each other. While using Sigmoid as an activation function the MSE (mean squared error) suffers from slow divergence, cross-entropy avoids this problem.

1.3.1.6 Dropout layer:

When the iteration is done overall training examples too many times, it results in overfitting. Dropout is a technique that prevents overfitting and provides a method of approximately combining exponentially many different neural network architectures efficiently [10]. Dropout layers do not have any parameters. They randomly set the previous layer's activation function to zero with a probability equal to the dropout ratio. Dropout is a process in which randomly selected neurons (units) are neglected during training. They are "dropped-out" randomly. So their contribution to the activation of downstream neurons is temporaly removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. Each unit is preserved with a fixed probability p independent of other units, where p can be chosen from a validation set or can simply be set at 0.5, commonly used dropout ratios for convolutional layers lies between 0.1 and 0.3.

1.3.1.7 Normalization:

During training, some pre-processing is performed on the input data. For example, all the data can be normalized so that data have zero mean and unit variance. Why there is a need for such pre-processing on the input data? The reasons being: avoiding the advance saturation of non-linear activation functions like sigmoid function, assuring that all input data is in the same range of values.

The distribution of the activation function is constantly changing during training within the intermediate layers. This effect is predominant while using the **ReLU** activation function, as it does not normalize its output. This slows down the training process because each layer must learn to adjust themselves to a new distribution in every training step. This problem is known as **internal covariate shift**. So Batch normalization is a method that we can use to normalize the inputs of each layer, in order to avoid the problem of internal covariate shift [11].

A batch normalization layer does the following operation during the training time:

- Calculate the mean and variance of the input of the layer.
- The layer inputs are normalized using the previously calculated batch statistics.
- Scaling and shifting are done to obtain the output of the layer.

1.4 COLOR SPACE

Color space is defined as a mathematical model that describes a range of colors. In color space, the color models are there that transforms the colors into numbers. Each color model is capable of producing a new color from a given set of primary colors. The range of color produced by any color model is defined as the color space. Different systems may use different color spaces according to their application. Five major color models are there:

- RGB
- CIE-LAB
- YUV/YCbCr
- HSL/HSV
- CMYK

In the proposed method we have used three color spaces: RGB, CIE-LAB, and YCbCr. So all three are discussed as follows:

The RGB color space is the most commonly used color space for processing and storing images in digital processing systems. By using a linear or non-linear transformation method any desired color space can be generated from RGB. It consists of three color components: red, green and blue. Any color can be obtained by mixing these three colors in different intensities. So they are called primary colors.

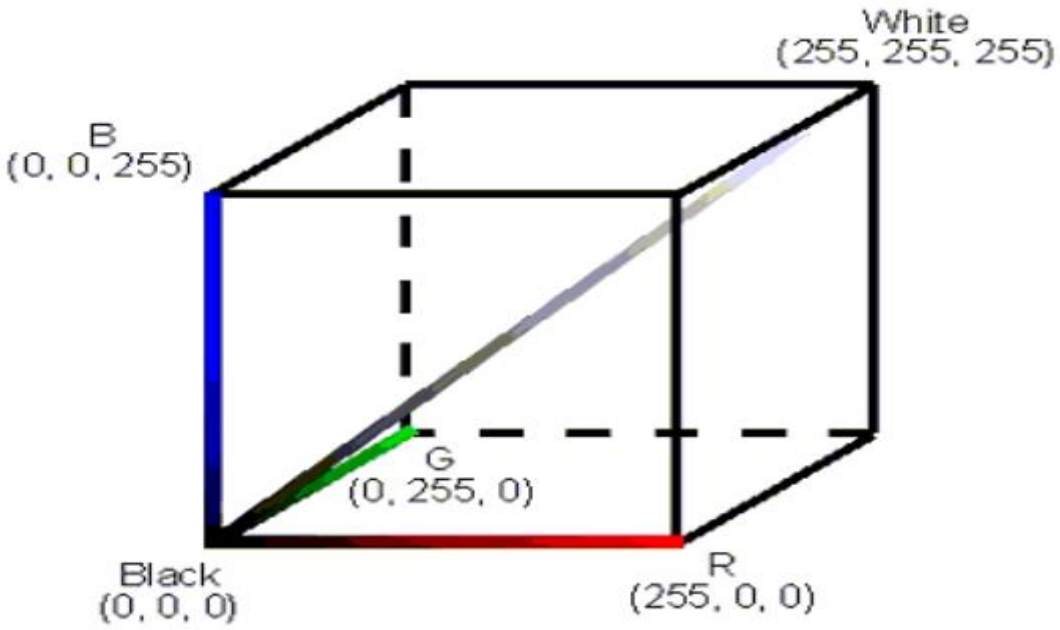


Figure 1.5 RGB color model [12]

Figure 1.5 represents the RGB color model. Here, coordinates (0,0,0) represents black color and (255,255,255) white color. In between the range of black and white color all other colors are present in the model.

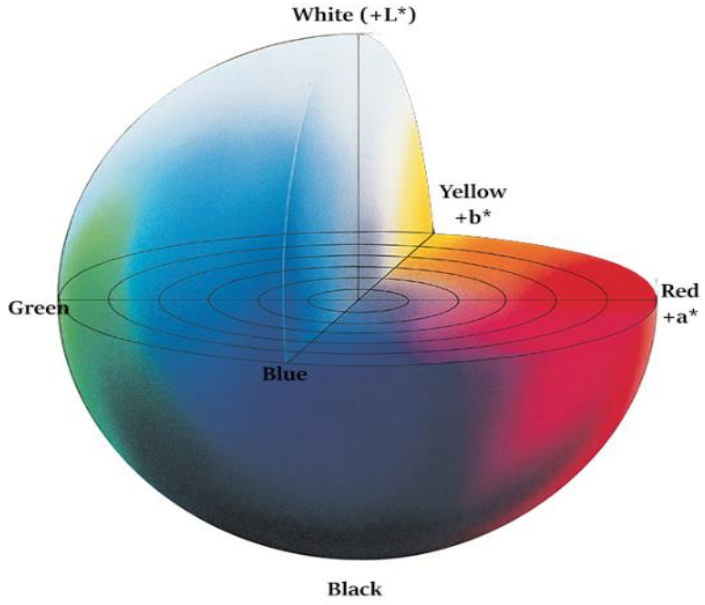


Figure 1.6 LAB color model [13]

Figure 1.6 shows the CIE-LAB color model. It represents three values: L for the lightness channel range from 0 (black) to 100 (white), a from green to red and b from blue to yellow. It provides three decorrelated channels, one principle channel which corresponds to the luminance value and the other two channels represent the chromatic values.

Nawar Fdhal et al. [14] discussed about the color space transformation RGB to CIE-LAB using neural networks. In this paper, the transformed results from the standard ICC profile and those from neural networks are compared. Pixels values in all practical color management systems are regulated by converting device-dependent values (RGB) into device-independent (CIE-LAB) values [15]. Basically, the device's color outcome is analyzed and modified and then this resulted data is used to estimate the transform relationship from RGB to LAB or vice-versa.

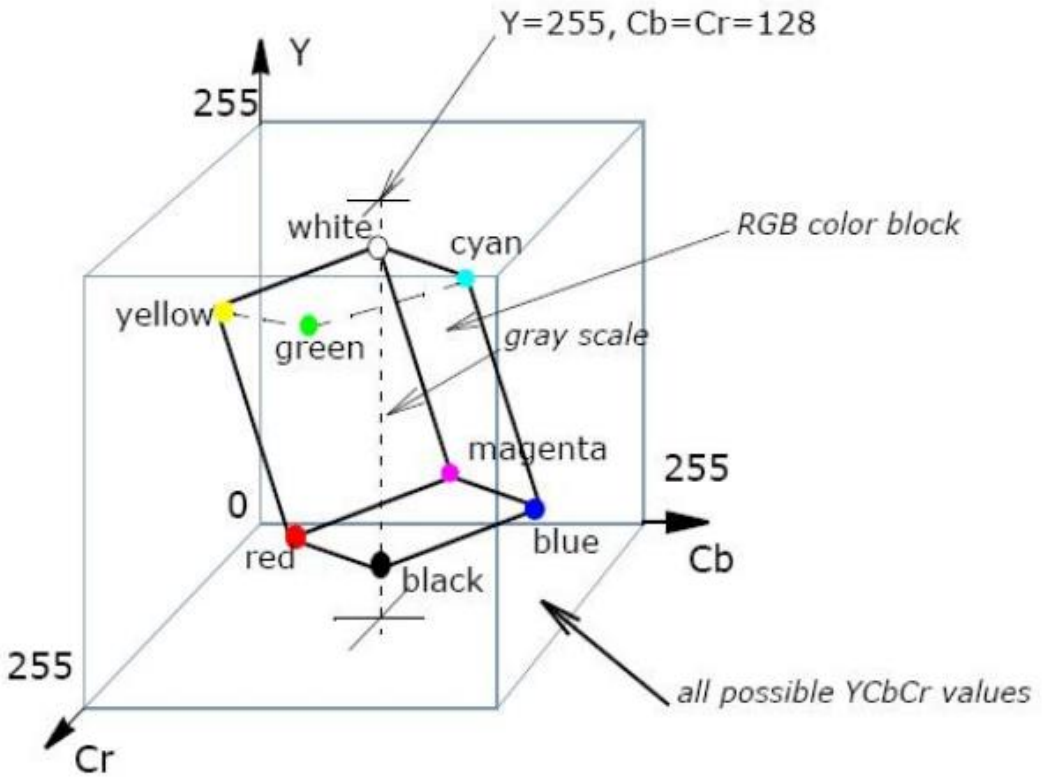


Figure 1.7 YCbCr color model [17]

In Figure 1.7 YCbCr color model represents three components: Y which is called the luma component defines the luminance and CbCr components that defines chromatic values. Luma is computed from a weighted sum of RGB values. Unlike RGB color spaceYCbCr color space is found as luminance independent so it provides better results. This color space is suitable for skin color segmentation [16].

1.5 PROBLEM STATEMENT

Luminance and chrominance are the two components that are represented by a color image. Any image can be reconstructed exactly if both of the components are given. The luminance image is a grayscale image that tells us about the object's edges and lighting effects in an image. The main purpose of colorization is to add the chromatic values to the grayscale image by regenerating the chromatic values for pixels in colors. So the actual problem is the regeneration of the chromatic values. We can estimate those values by using a reference color image. The relationship between the color image and the grayscale image is examined, depending on that relation transformation is done. The color spaces used in this work are LAB and YCbCr color space. We convert RGB to LAB color space. A Lab encoded image has one layer of a grayscale image. Grayscale layer act as input to the network and two-color layers are predicted through the network as output. The goal of this thesis is to find an improved method for the automatic colorization of grayscale images.

CHAPTER 2

GENERATIVE ADVERSARIAL NETWORK

2.1 INTRODUCTION TO GENERATIVE ADVERSARIAL NETWORK

Generative Adversarial Networks (GANs) are a powerful class of deep neural network that is used for unsupervised learning. It was developed and introduced by Ian J. Goodfellow in 2014. It consists of two competing networks: a generator network and a discriminator network. Both networks work together to provide a high-level simulation of the tasks. It offers a lot of potential in the world of artificial intelligence.

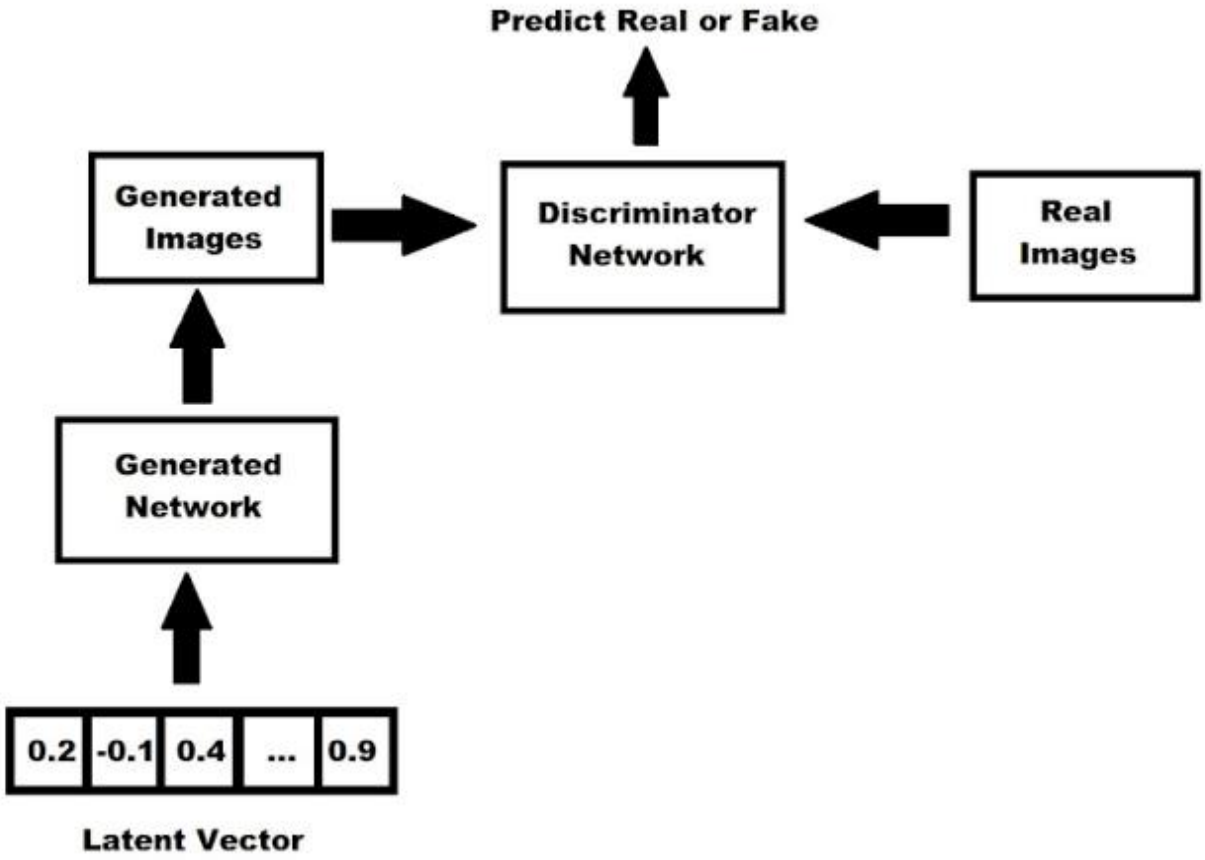


Figure 2.1 A GAN network [18]

Figure 2.1 shows the basic architecture of a GAN network. The function of the generator is to generate data that resembles the training data. The generated data can be an image or audio. It is not easy to measure how much a generated image looks like the training set. The discriminator is trained so well to tell what's the probability that the generated image belongs to the training set class. So Generator generates fake samples of data and tries to fool the Discriminator. The Generator and the Discriminator run in competition with each other in the training phase. After a few iterations, the generator and discriminator get better and better in their respective job. The goal of a GAN network can be described through a min-max game. The Discriminator is trying to minimize its reward **V(D, G)** and the Generator is trying to minimize the Discriminator's reward or in other words, maximize its loss. Mathematically it can be represented as:

$$min_G \, max_D \mathrm{V(D,G)}$$

$$V(D,G) = E_{x \sim Pdata(x)}[logD(x)] + E_{z \sim P(z)}[log\ (1 - D\big(G(z)\big)]$$

Where,

D= Discriminator

G= Generator

G(z)= Generator network

D(x)= Discriminator network

P(z)= Distribution of generator

Pdata(x)=Distribution of the real data

x= Sample from Pdata(x)

z=Sample from P(z)

2.2 GENERATIVE vs. DISCRIMINATIVE ALGORITHMS

To understand GANs we must know how generative and discriminative algorithms work. Discriminative algorithms try to characterize input data and predict a label or category to which that data belongs. Discriminative algorithms map features to labels. For example, if all the words in an email are given, a discriminative algorithm predicts whether the message is spam or not

spam. Spam is one of the labels, and the words from the email are the features that represent the input data. Mathematically, if the label is represented by 'y' and the feature is represented by 'x'. Then P(y/x) defines the probability of y given x (the probability that an email is spam given the words it contains). The generative algorithm does the opposite of the discriminative algorithm. Generative models framework the distribution of individual classes. In this case, the probability is defined as P(x,y) =P(x/y)P(y), which models the actual distribution of the data. P(x/y) defines the probability of x given y. In joint probability distribution P(x,y), given a y, we can calculate x. So these models are called "generative" models.

2.3 TRAINING OF A GAN

Training a GAN is performed mainly in two steps:

Step 1: Generator is kept inactive and the discriminator is under training. In this step, the network is propagated in the forward direction only no back-propagation is done. For n number of epochs, the discriminator network is trained on real data to check whether it can predict the real data or not. The discriminator is trained on the data generated by the generator.

Step 2: In this step discriminator is kept idle and the generator network is under the training process. Once the discriminator is trained on the data generated by the generator then we get the predictions. We use these results for training the generator and get better from the previous state to try and fool the Discriminator. The above step is repeated for a few epochs and we check the fake data if it seems genuine. If it is considerable, then the training is stopped.

2.4 TYPES OF GANs

There have been many different types of GAN implementation. Some of them are described below:

1. Vanilla GAN: This is the simplest type of GAN. The Generator and the Discriminator network are simple multi-layer perceptrons. In vanilla GAN, the algorithm is really simple. It makes use of stochastic gradient descent to optimize the mathematical equation.

2. Deep Convolutional GAN (DC-GAN): DC-GAN is one of the most successful implementations of GAN. It is composed of Convolutional networks in place of multi-layer perceptrons. It replaces all max pooling with convolutional stride. It eliminates fully connected layers within the network.

3. Conditional GAN (C-GAN): In the C-GAN, we put a condition on the generator as well as discriminator model. Figure 2.2 shows the architecture of a C-GAN. Here, y is given as a piece of extra information. It can be any kind of information such as a class label. In the generator model of a C-GAN, the prior input noise and auxiliary information y are combined in a joint hidden representation.

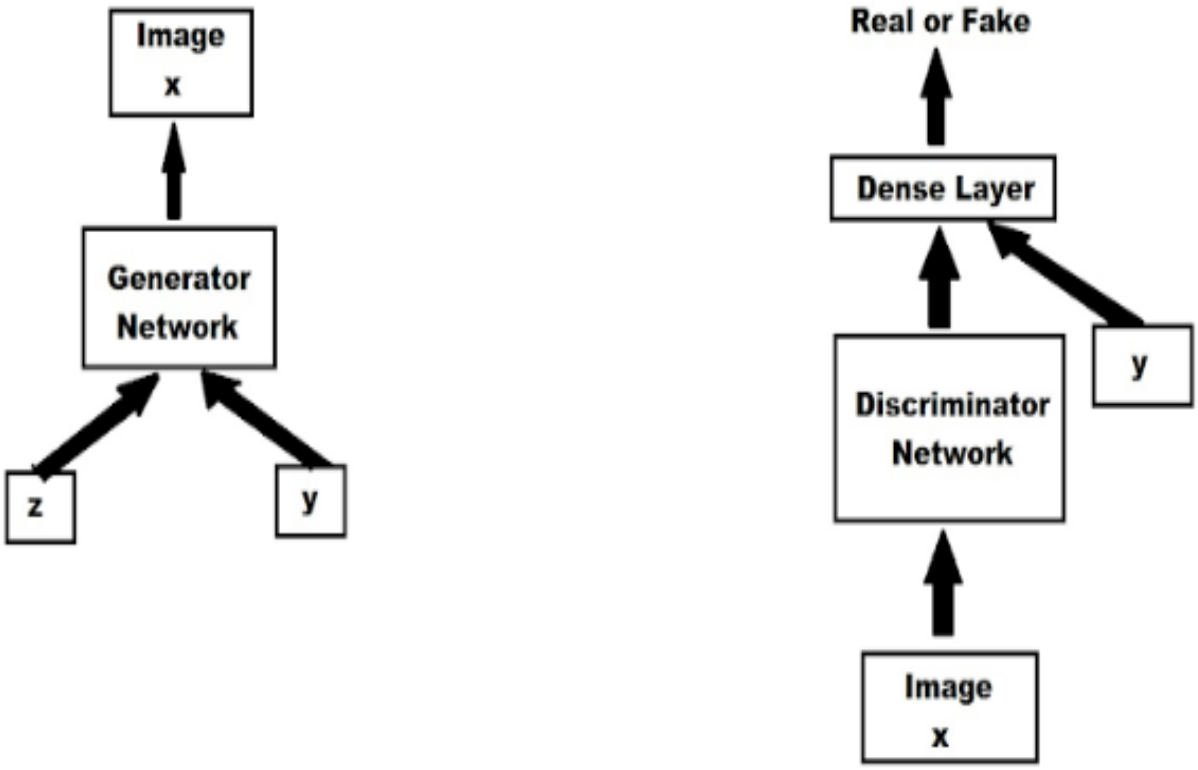


Figure 2.2 Conditional Generative Adversarial Network [19]

4. Laplacian Pyramid GAN (LAP-GAN): In the Laplacian pyramid image is represented in a linear invertible form, it consists of a set of band-pass images and spaced by an octave apart. This type of GAN uses multiple numbers of generator and discriminator networks and different levels of the Laplacian Pyramid. This approach produces very high-quality images. Downsampling of the input image is done at each layer in the pyramid network and during the backward pass upscaling is performed.

5. Super Resolution GAN (SR-GAN): In SR-GAN a deep neural network is used along with an adversarial network to get higher resolution images. This design is useful for improving the resolution of the images.

CHAPTER 3

LITERATURE REVIEW

3.1 COLORIZATION OF GRAYSCALE IMAGES

Over the last decade, the automatic colorization of the grayscale images is an appealing area in the field of image processing**.** It has a major application in the restoring of aged or degraded images. Color is added into a grayscale image to improve the visual appeal of the image. Different authors have proposed different methods for the colorization each of them describing their own method. Convolutional neural networks (CNN), Deep CNN, Generative adversarial network (GAN) and DC-GAN, these are some of the architecture that has been used in the various papers. While comparing CNN and GAN results for colorization it is observed that the images colored by using the GAN model have high quality in terms of brightness and sharpness.

Ian J. Goodfellow et al. [3] developed a powerful class of neural networks called Generative Adversarial Network in 2014. He proposed a method for estimating the generative models using an adversarial process. GAN consists of two models: the generator model and the discriminator model. The generator model defines the data distribution while the discriminator model estimates the probability of the sample data whether it came from the actual dataset rather than the generator's output. Both the generator and discriminator have their own goal during the training process. The generator tries to maximize the discriminator's probability of making a mistake. Forward propagation, backpropagation, and dropout algorithm are used for training the network. This method removes the need for the Markov chain or any approximation inference networks required for the generation of samples.

Phillip Isola et al. [5] defined a conditional adversarial network that is used for the various image to image translational problems. The conditional adversarial network puts a condition on the generative model to produce corresponding output data. This makes C-GAN useful for the various image to image translation tasks. This network learns a mapping between the input image and the output image. It also learns a loss function that is used during the training procedure of the mapping. The design of the generator model used in this paper is a "U-net" based architecture and for discriminator model, a convolutional "PatchGAN" classifier is used. The tasks which mainly require highly structured graphical output make use of conditional adversarial networks.

Nitish Srivastava et al. [11] described a method to prevent neural networks from the problem of getting overfitted. A neural network has very complex relationships between its inputs and outputs. Overfitting occurs when the iteration is done overall training examples too many times. Dropout is a technique that prevents overfitting and provides a method of approximately combining exponentially many different neural network architectures efficiently. This improves the performance of a neural network. It regularizes the neural networks by adding noise to their hidden units. Five different datasets SVHN, CIFAR-100, CIFAR-10, MNIST, and Imagetnet have been chosen to evaluate the dropout technique. This technique is also applied in a speech recognition task. TIMIT dataset is used during the process. Without dropout, a six-layer network gives a phone error rate of 23.4 %, but by using dropout in the architecture it improved to 21.8%.

Sergey Ioffe et al. [12] represented the technique of batch normalization. Training of neural networks is a very complicated task. During the training process, the previous layer's parameters get changed due to which input distribution for the next layer gets changes. This slows down the training process as it requires careful initialization of the parameters and a lower learning rate. This problem is named as internal covariance shift. So batch normalization is used to overcome this problem. Once the problem is overcome the training process gets faster.

Richard Zhang et al. [20] proposed a method for converting a grayscale image into a colored image using a deep convolutional neural network. He developed a fully automatic technique that gives a realistic colorization. In this paper, the problem is taken as a classification task and class rebalancing is used during the training time to increase the diversity of colors in the final output. A "colorization Turing Test" is done in which humans are asked to participate and are asked to choose between a generated sample and a ground truth sample. This method successfully fools humans up to 32% of the trial. Colorization is explored as a self-supervised learning problem. The L channel of the input image is considered as input and it’s AB channels as the supervisory signal.

Anat Levin et al. [21] proposed a simple method for the colorization of images. In colorization, the major problem is that the process is expensive and time-consuming. The image is segmented into several regions and then the color is assigned to each of the region. The technique proposed in this paper is based on the theory that if nearby pixels in space-time have the same intensities or gray level then they should have the same color. A cost function is used to obtain an optimized problem. This method of colorization doesn't require any image segmentation or accurate

tracking of regions. The input image is defined with a few color scribbles only, the provided color propagate through space and time both to produce a colored image. This method reduces the amount of input data required from the user. This method is applicable to selective recoloring, a useful operation in digital photography.

Alec Radford et al. [22] represented a class of the generative adversarial network called a deep convolutional generative adversarial network. This DC-GAN has some structural constraints but it fills the gap between the success of CNN architecture for supervised and unsupervised learning.

Tomihisa welsh et al. [23] represented a technique for colorizing the grayscale image by transferring the color between the source image and target image. The source image is a color image and the target image is the grayscale image. The proposed method tends to minimize the human guidance required for the task. By matching the luminance and texture information between the images, we transfer the whole color "mood" of the source image to the target image. Initially, the desired color mood of a particular swatch region in the source image is transferred to the corresponding swatch region of the target image. During the final stages of the colorization process, by using a texture synthesis colored swatches are employed and then the remaining part of the grayscale image is colored.

Mehdi Mirza et al. [24] represented the generative adversarial network that has removed the need for Markov chains. In GAN only backpropagation is required to obtain the gradients. This paper mainly focuses on the conditional adversarial network. In conditional GAN we give some additional data or information to the generative model. By putting a condition on both the generator and the discriminator we have control on the generated data. The applied condition or the given information can be a class label or data from the other modalities.

Stephen Koo et al. [25] has represented a deep convolutional generative adversarial network (DC-GAN) technique for automatic colorization of black and white photos. A baseline model is designed using a feed-forward convolutional neural network. This model is trained on the CIFAR-10 dataset. This baseline model is reformulated as a generator model that takes a grayscale image and noise as input and generates the output, a discriminator model is trained over the real and fake both images so that it can predict that whether the input provided is from the generated sample or from the actual dataset. In this paper, YUV color space has been used instead of RGB color space, as YUV minimizes the per-pixel correlation between the color

channels. The proposed method takes the Y component as the input and predicts the UV components as output. So basically to get the advantage of the generative models with deep convolutional neural networks, he first designed a baseline architecture or model and then incorporate this model into an adversarial framework.

Jeff Hwang et al. [26] proposed a technique for colorization that uses a statistical learning approach. A convolutional neural network is designed in which a black and white image is fed as an input and a colored image is obtained as an output. Whatever the system has learned from its past based on that learning it produces the output images. In this paper, a comparison between the classification network and regression network is done. The results of a regression network are slightly valid. The resulted images are desaturated and unappealing. In contrast, the output of a classification network is realistic. Colors are saturated and tightly restricted to the region they correspond to. So finally it is concluded that the images generated by a classification model are more pleasing than the images generated by a baseline regression model.

Yun Cao et al. [27] represented a method that uses unsupervised learning in the conditional generative adversarial network for diverse colorization of grayscale images. Unlike other GAN architectures that use convolutional and deconvolution layers as encoder and decoder respectively, a fully convolutional generator has been used in this paper. The conditional information is interleaved in each layer using the concatenate layer.

Xavier Glorot et al. [28] discussed about the difficulties that occur during the training of the neural networks and the role of nonlinear activation function in the training process. Without an activation function, our network becomes a linear regression model that has limited power. A sigmoid activation function makes the hidden layers to fall into the saturation. So the networks that use sigmoid function converges very slowly. This paper discussed about the various activation functions each having its own effect over the hidden layers.

Raj Kumar Gupta et al. [29] represented the automatic colorization technique. In this paper, the example-based method is used for colorization. The basic goal of a colorization method is to add colors to the input image to make it meaningful. A method is proposed which extracts various features from the image and uses these features to transfer the color information from source image to target image. SURF and Gabor features are used. The proposed method uses superpixel's resolution and the similarities between these superpixels are find out by using a

matching method. Various features are check over each step of matching. This introduces spatial consistency in the colorization process.

Viren Jain et al. [30] represented a procedure for denoising of natural images by using a convolutional neural network and an unsupervised learning method. After training the proposed model over a given dataset it is found that the CNN model performs best over the wavelet and Markov chain method.

Bogdan Mazoure et al. [31] proposed an image to emoji architecture using GAN. The proposed architecture is trained over various social datasets. In this paper, the distribution of emoji is formed which is conditioned on an image using a deep generative architecture. These GAN are difficult to train than the other models like auto encoder-decoder but they produce powerful results in various computer vision tasks.

Christian Ledig et al. [32] represented SRGAN, a generative adversarial network specially designed for image superresolution. This is the first model that can predict photo-realistic images for 4x upscaling factors. To attain this, a perceptual loss function is proposed that is comprised of an adversarial loss and a content loss. A discriminator model is trained to distinguish between the ground truth image and the super-resolved image. The proposed deep residual network is capable of recovering the photo-realistic texture from deeply downsampled images.

Patricia L. Suarez et al. [33] represented a novel technique for colorization of NIR images by using DC-GAN. For the learning process of each channel separately triplet model is used. This results in rapid convergence during the training period. A large degree of similarity is obtained between the colored NIR images and the original images.

Ayush Jaiswal et al. [34] represented the Bidirectional conditional generative adversarial model. In conditional GAN the data samples (u) are generated having conditioned on both the latent variable (y) and the known information (z). The proposed BiCoGAN approach easily untangle the y and z in the generation process and inverse mapping from u to both y and z is learned via an encoder. In this paper, a vital technique for the training of the BiCoGANs is presented that employs an extrinsic factor loss and a dynamically tuned weight. While comparing with other encoder based C-GAN, BiCoGAN encodes more precisely and uses y and z in a more efficient manner to generate accurate results.

3.2 IMAGE COLORIZATION IN MEDICAL FIELD

GAN plays a very important role in the medical field also. They are used for the colorization purpose or as a detector. Most of the medical equipment reproduces their output in grayscale format. A human eye is incapable of distinguishing more than a few hues of gray, so here the colorization process is very helpful.

Olaf Ronneberger et al. [35] discussed the role of a convolutional neural network in biomedical imaging. In this paper, the U-net structure is used to perform different segmentation problems like cell segmentation and neuronal structure segmentation. The segmentation of neuronal structure is done using electron microscopic recording. In a cell, segmentation recording is done by using phase-contrast microscopy. In the proposed data augmentation for very few annoted training images are required during training that reduces the training period significantly.

Xin Yi et al. [36] represented the application of GAN in medical imaging. This paper describes the different types of GAN used for different medical applications. As generative networks are capable of learning various features from the training data and producing new images, it is used to imitate the data. The discriminator network works as a detector. First, it is trained over the training examples then it detects which one is real and which one is fake. This paper shows how GAN is used for medical purposes like multimodal or unimodal image registration, image reconstruction, cell segmentation, image classification, and detection.

Houssam Zenati et al. [37] proposed a method for anomaly detection. For the simulation of complex datasets and high dimensional data, GANs are very effective. In this paper, the proposed BiGAN (bidirectional GAN) method achieves the results with a high degree of accuracy. This method takes less time during training. MNIST dataset and a network dataset KD99 is used in the paper.

Karim Armanious et al. [38] discussed about the MedGAN that is used especially for medical image-to-image translation. MedGAN is a field of the generative adversarial network that is reconstructed by merging non-adversarial losses with an adversarial model. In this paper, the discriminator model is defined as a trainable feature extractor. The divergence between the resulted image and the desired modalities is corrected by this discriminator model. The resulted looses are used to match the texture and fine-structure of the translated image with the desired images. In this paper, a new generator architecture CasNet is proposed that compromises of

various encoder-decoder pairs which improves the acuteness of the translated medical output. The proposed MedGAN architecture is applied to three different tasks: deblurring of PET images, PETCT translation, and correction of MR motion artifacts. On the basis of the various perceptual analysis, it is concluded that MedGAN overreaches the other translation methods.

Qingsong Yang et al. [39] proposed a denoising method for medical images using the generative adversarial network. The widespread use of CT in the medical field has upraised an open concern on the amount of radiation dose given to the patient. Reduction in the amount of the dose to the patient will result in increased noise and artifacts. This paper basically introduces a generative adversarial model with Wasserstein's distance for denoising the CT images. Wasserstein distance enhances the performance of the GAN. The perceptual loss restrains the noise by comparing the features of a denoised image with the ground truth image in an ascertained feature space. The noise distribution of the data is migrated from strong to weak statics using the proposed GAN model. The proposed technique provides powerful results in experiments with CT images.

Dan C. Ciresan et al. [40] proposed a method to discover the mitosis in breast cancer histology pictures. The number of mitosis figures in the pictures of a breast is important for cancer screening. The counting of these figures can be done manually. In this paper, a convolutional neural network with maximum pooling has been used for the detection purpose. Each pixel in the images is classified by training the network.

CHAPTER 4

TRAINING OF THE GAN

4.1 INTRODUCTION

In this work, a GAN network has been built in Keras. The type of GAN structure used in the work is conditional GAN. Conditional GAN as discussed above are a type of GAN in which some extra information is fed to the generator and discriminator both. This condition can be a class label. Based on that condition the data is generated through the model. The whole procedure consists of different parts like building the generator and discriminator model their training and all.

4.1.1 Importing packages:

```
from keras.models import Model

fromkeras.layersimportConv2D,MaxPooling2D,Activation,BatchNormalization,UpSampli
ng2D,Dropout,Flatten,Dense,Input, LeakyReLU, Conv2DTranspose, AveragePooling2D,
Concatenate

from keras.optimizers import Adam

from keras.layers.advanced_activations import LeakyReLU

from keras.models import Sequential

from tensorflow import set_random_seed

from keras.utils import np_utils

import h5py

import numpy as np

from numpy import *

import os

from PIL import Image

from PIL import ImageFilter

from sklearn.utils import shuffle

from sklearn.model_selection import train_test_split

import matplotlib

from skimage import color,io

from keras.models import load_model

from keras.layers import LeakyReLU
```

```
import matplotlib.pyplot as plt
import keras.backend as K
np.random.seed(1)
set_random_seed(1)
```

As we have built our model in Keras so we are importing various packages like layers, models, optimizer and activation functions.

4.1.2 Reading images from the dataset:

```
imRG=array([array(Image.open(path1+'/'+im6)) for im6 in list_i],'f')
```

The dataset contains images of size 256x256x3 from a different class. The images are read in the form of an array.

4.1.3 Training and testing set:

```
label=np.ones((s,),dtype=int)
label[0:]=0
data,Label= shuffle(imRG,label,random_state=2)
train_data= [data,Label]
print(train_data[0].shape)
(u)=(train_data[0])
#split dataset into training and testing set....
x_train,x_test=train_test_split(u,test_size=0.3)
x_train.shape
```

In this section, the dataset is split into training and testing set. *Test_size* here defines what percentage of the dataset should be allocated to testset.

4.1.4 RGB to LAB/YCbCr:

The images in the dataset are in RGB format. In this section, we will convert the images from RGB to LAB color space. In LAB color space L represents the grayscale channel and AB

represents the two chromatic channels. LAB color space reduces the correlation between the three axes. So it provides three decorrelated channels, one principle channel which corresponds to the luminance value and the other two channels represent the chromatic values. Because the channels are decorrelated with each other so any change within the one channel will have a nominal effect on the other two channels. So color transformation gets easy as there is no cross-channel artifact. The GAN architecture is also trained on the pictures in the YCbCr color space. In this color space Y represents the luma component. So we feed Y channel to the generator and in result get CbCr channel. When compared to RGB color space, YCbCr color space is independent of the luminance component.

4.1.5 Generator model:

```
gen_input = Input(shape=self.g_input_shape)
conv1 = Conv2D(64, (3, 3), padding='same', strides=2)(gen_input)
conv1=LeakyReLU(alpha=0.2)(conv1)
conv1 = BatchNormalization()(conv1)
conv2 = Conv2D(128, (3, 3), padding='same', strides=2)(conv1)
conv2=LeakyReLU(alpha=0.2)(conv2)
conv2 = BatchNormalization()(conv2)
conv4 = Conv2D(256, (3, 3), padding='same', strides=2)(conv2)
conv4=LeakyReLU(alpha=0.2) (conv4)
conv4 = BatchNormalization()(conv4)
conv5 = Conv2D(512, (3, 3), padding='same', strides=2)(conv4)
conv5 = BatchNormalization()(conv5)
conv5 =LeakyReLU(alpha=0.2)(conv5)
conv6 = Conv2D(512, (3, 3), padding='same', strides=2)(conv5)
conv6=LeakyReLU(alpha=0.2)(conv6)
conv6 = BatchNormalization()(conv6)
conv7 = Conv2D(512, (3, 3), padding='same', strides=2)(conv6)
conv7 = BatchNormalization()(conv7)
conv7=LeakyReLU(alpha=0.2)(conv7)
conv8 = Conv2D(512, (3, 3), padding='same', strides=2)(conv7)
conv8 = BatchNormalization()(conv8)
```

```
conv8 = LeakyReLU(alpha=0.2)(conv8)

conv9 = UpSampling2D(size=(2, 2))(conv8)

conv9 = Conv2D(512, (3, 3), padding='same',strides=1)(conv9)

conv9 = BatchNormalization()(conv9)

conv9 = Activation('relu')(conv9)

conv9 = Concatenate(axis=-1)([conv9,conv7])

conv10 = Conv2D(512, (3, 3), padding='same') (conv9)

conv10 = BatchNormalization()(conv10)

conv10 = Activation('relu')(conv10)

up2 = UpSampling2D(size=(2, 2))(conv10)

conv11 = Conv2D(512, (3,3), padding='same',strides=1)(up2)

conv11 = BatchNormalization()(conv11)

conv11 = Activation('relu')(conv11)

conv11 = Concatenate(axis=-1)([conv11,conv6]

conv12 = Conv2D(512, (3, 3), padding='same')(conv11)

conv12 = BatchNormalization()(conv11)

conv12 = Activation('relu')(conv12)

up3 = UpSampling2D(size=(2, 2))(conv12)

conv13 = Conv2D(512, (3,3), padding='same',strides=1)(up3)

conv13 = BatchNormalization()(conv13)

conv13 = Activation('relu')(conv13)

conv13 = Concatenate(axis=-1)([conv13,conv5])

conv14 = Conv2D(512, (3, 3), padding='same')(conv13)

conv14 = BatchNormalization()(conv14)

conv14 = Activation('relu')(conv14)

up4 = UpSampling2D(size=(2, 2))(conv14)

conv15 = Conv2D(256, (3, 3), padding='same',strides=1)(up4)

conv15 = BatchNormalization()(conv15)

conv15 = Activation('relu')(conv15)

conv15 = Concatenate(axis=-1)([conv15,conv4])

conv16 = Conv2D(256, (3, 3), padding='same')(conv15)

conv16 = BatchNormalization()(conv16)

conv16 = Activation('relu')(conv16)
```

```
up5 = UpSampling2D(size=(2, 2))(conv16)
conv17 = Conv2D(128, (3, 3), padding='same',strides=1)(up5)
conv17 = BatchNormalization()(conv17)
conv17 = Activation('relu')(conv17)
conv17 = Concatenate(axis=-1)([conv17,conv2])
conv18 = Conv2D(128, (3, 3), padding='same') (conv17)
conv18 = BatchNormalization()(conv18)
conv18= Activation('relu')(conv18)
up6 = UpSampling2D(size=(2, 2))(conv18)
conv19 = Conv2D(64, (3, 3), padding='same',strides=1)(up6)
conv19 = BatchNormalization()(conv19)
conv19 = Activation('relu')(conv19)
conv19 = Concatenate(axis=-1)([conv19,conv1])
conv20 = Conv2D(64, (3, 3), padding='same')(conv19)
conv20= BatchNormalization()(conv20)
conv20 = Activation('relu')(conv20)
up7 = UpSampling2D(size=(2, 2))(conv20)
conv21 = Conv2D(2, (3, 3), padding='same',strides=1)(up7)
conv21 = Activation('tanh')(conv21)
model = Model(inputs=g_input,outputs=conv21)
return model
```

The purpose of the generator model is to generate data that resembles the original data. So after converting the images into LAB/YCbCr color space, we get separate L/Y channel and AB /CbCr channel. The shape of the images in L/Y channels is like 256x256x1 and in AB/CbCr channel 256x256x2. The input given to the generator model is in the shape 256x256x1 means we are feeding L/Y channel into the generator and in return, we want AB/CbCr channel. So the output of the generator model must have shape 256x256x2. According to this, we set up several hidden layers in the network each layer having its different filter size. At starting the size of the image is downsampled by using strides=2. Then again upsampling is performed to get the desired shape. So here the purpose of the generator model is to generate the AB channel from the given L channel. Figure 4.1 represents a flow chart of a generator model.

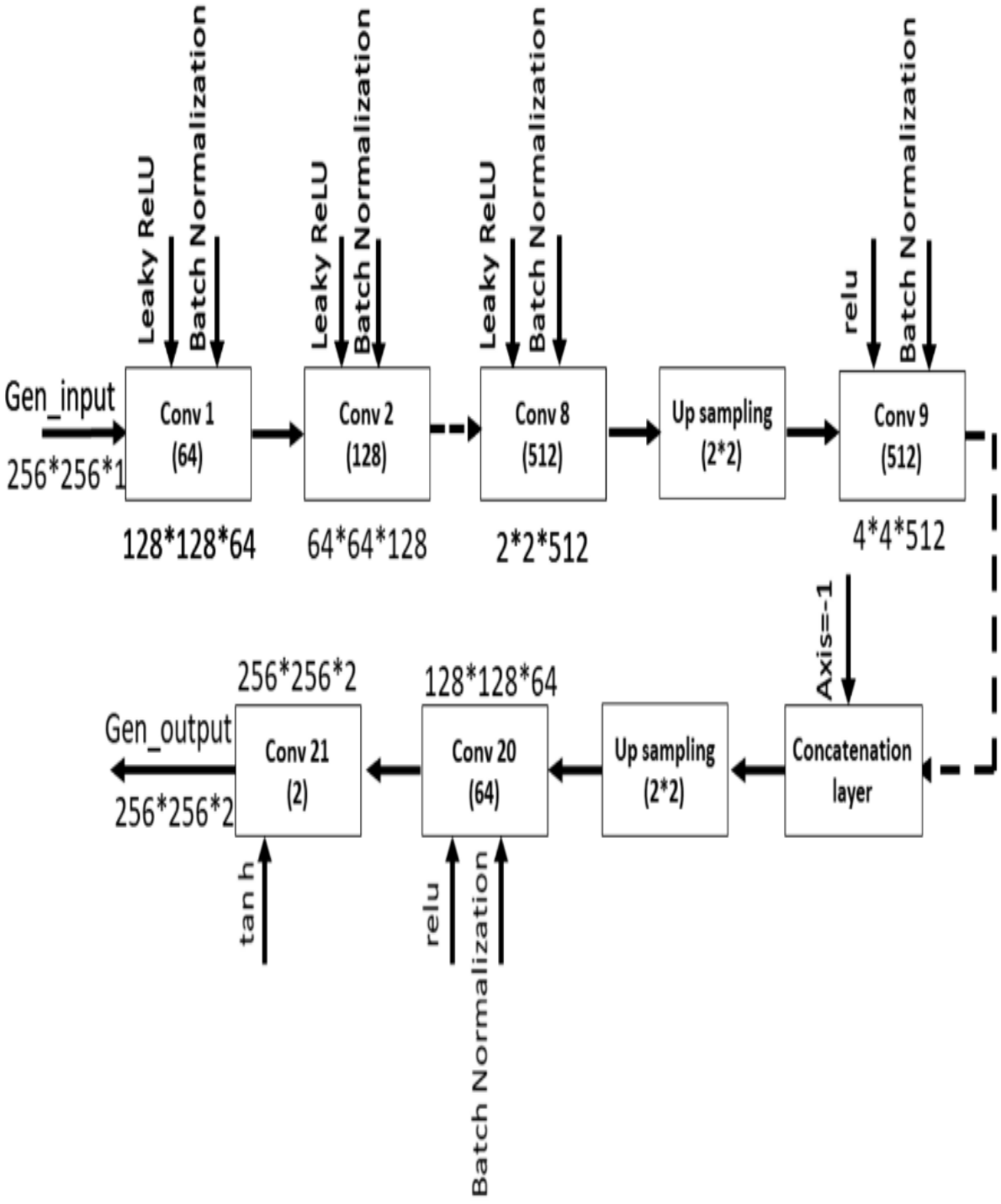


Figure 4.1 Generator model

4.1.6 Discriminator model:

```
model = Sequential()
model.add(Conv2D(32,(3,3),padding='same',input_shape=self.d_input_shape, strides=2))
model.add(LeakyReLU(.2))
model.add(Dropout(.3))
model.add(Conv2D(64, (3, 3), padding='same',strides=2))
model.add(BatchNormalization())
model.add(LeakyReLU(.2))
model.add(Dropout(.3))
model.add(Conv2D(128, (3, 3), padding='same',strides=2))
model.add(BatchNormalization())
model.add(LeakyReLU(.2))
model.add(Dropout(.3))
model.add(Conv2D(256, (3, 3), padding='same',strides=2))
model.add(BatchNormalization())
model.add(LeakyReLU(.2))
model.add(Dropout(.3))
model.add(Flatten())
model.add(Dropout(.3))
model.add(Dense(1))
model.add(Activation('sigmoid')
 return model
```

The first layer of the discriminator receives the input with shape 256x256x2. The discriminator also consists of different hidden layers. The output of the discriminator tells whether the data sample is real or fake.

4.1.7 GAN:

In this section compiling of both the generator and discriminator is done separately and they are formed as a single model called GAN. During training, the network Adam optimization is used. This algorithm is easy to implement and requires less memory to do complex operations.

4.1.8 Training:

Training of GAN begins with the training of discriminator. The discriminator network is trained on real data to check whether it can predict the real data or not. Further, the discriminator is trained on the data generated by the generator. During the training of discriminator, the generator is kept in idle state. Once the discriminator is trained over both real and generated data then it can make the prediction. Now the GAN is trained over the grayscale images and results in the colored output in the LAB/YCbCr color space. Accuracy of the network is maintained by calling *train_discriminator* function when the accuracy falls below some specified value. As the type of GAN used in this work is the conditional GAN. During the training time, the additional information or the condition according to which output has to be generated is added to both generator and discriminator. The GAN is conditioned on the grayscale image which is the input to the generator.

CHAPTER 5

RESULTS AND DISCUSSION

After training our model with two color spaces we obtained two generator models. Model *'gen1_model'* represents the generator model with LAB color space and *'gen2_model'* represents the generator model with YCbCr color space.

```
gen1_model=load_model('/content/drive/My Drive/models/uma1g40.h5')

gen2_model=load_model('/content/drive/My Drive/models/umaaimg40.h5')

img_lst_pred = model.predict(X_L/Y)
```

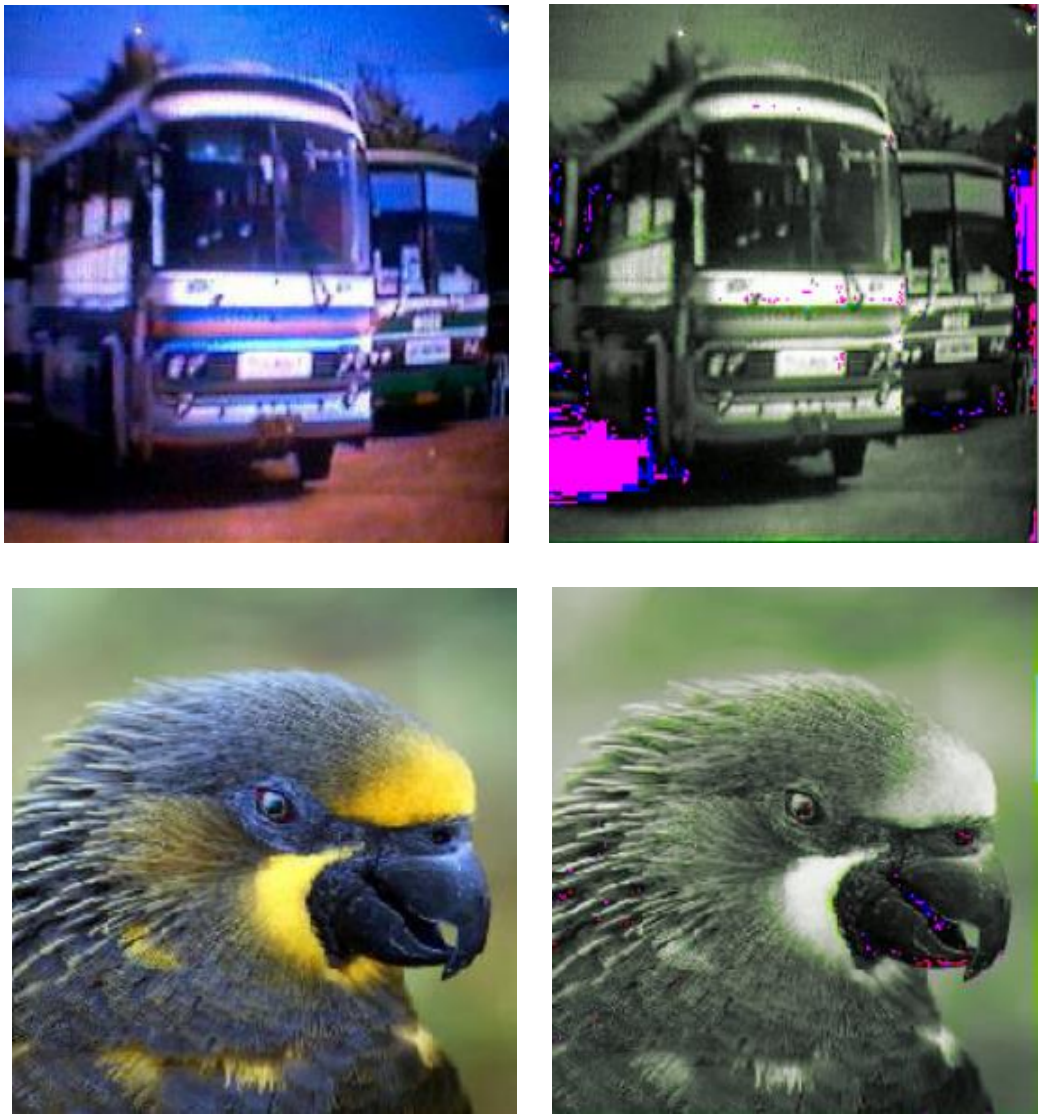
The *load_model()* function is used to load the trained generator model from the drive. The *model.predict()* function generators the samples using the generator model. In the case of LAB color space predicted sample corresponds to the AB channel and in YCbCr color space it corresponds to the CbCr channel. After getting the predicted channel, we concatenate these channels with the grayscale/luminance channel. A predicted AB/CbCr channel is concatenated with L/Y channel to get the generated image and an actual AB/CbCr channel is concatenated with L/Y channel to get the original image

5.1 RESULTS USING LAB COLOR SPACE

(a) **(b)**

Figure 5.1 Results with LAB color space (a) Original image (b) Image generated using GAN

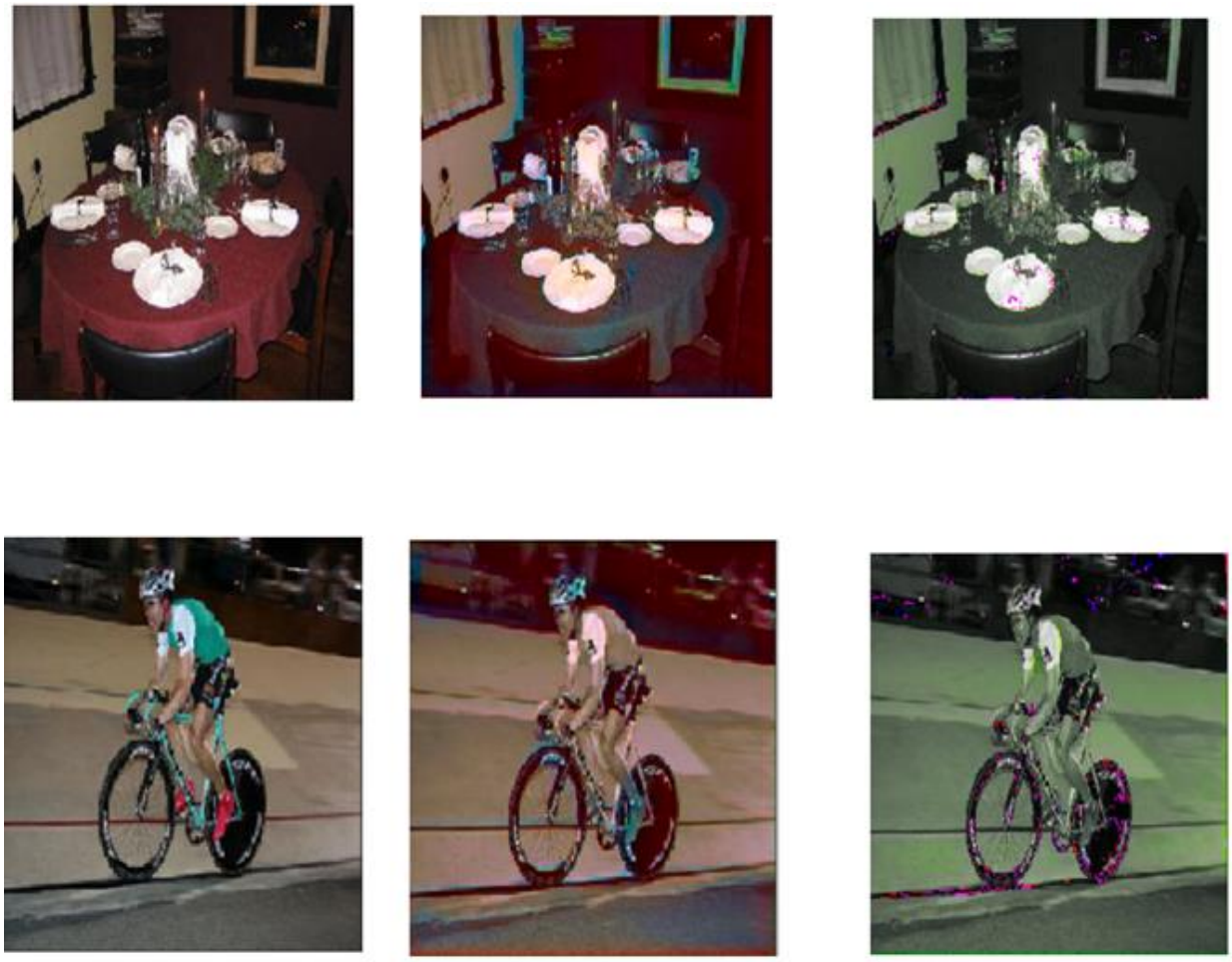
5.2 RESULTS USING YCbCr COLOR SPACE



(a) **(b)**

Figure 5.2 Results with YCbCr color space (a) Original image (b) Image generated using GAN

5.3 COMPARISON OF RESULTS FROM BOTH COLOR SPACES



(a) **(b)** **(c)**

Figure 5.3 Comparison of generated images (a) Real image (b) GAN output using CIE-LAB color space (c) GAN output using YCbCr color space

Figure 5.3 shows the comparisons between the images generated by the GAN using LAB color space and YCbCr color space. The images generated in LAB color space has more color information as compared to the images generated in YCbCr color space. The resulting images are not matching exactly with the original images in both (b) and (c) cases but the images with LAB color space are good in brightness and sharpness level.

5.4 MODEL GRAPHS

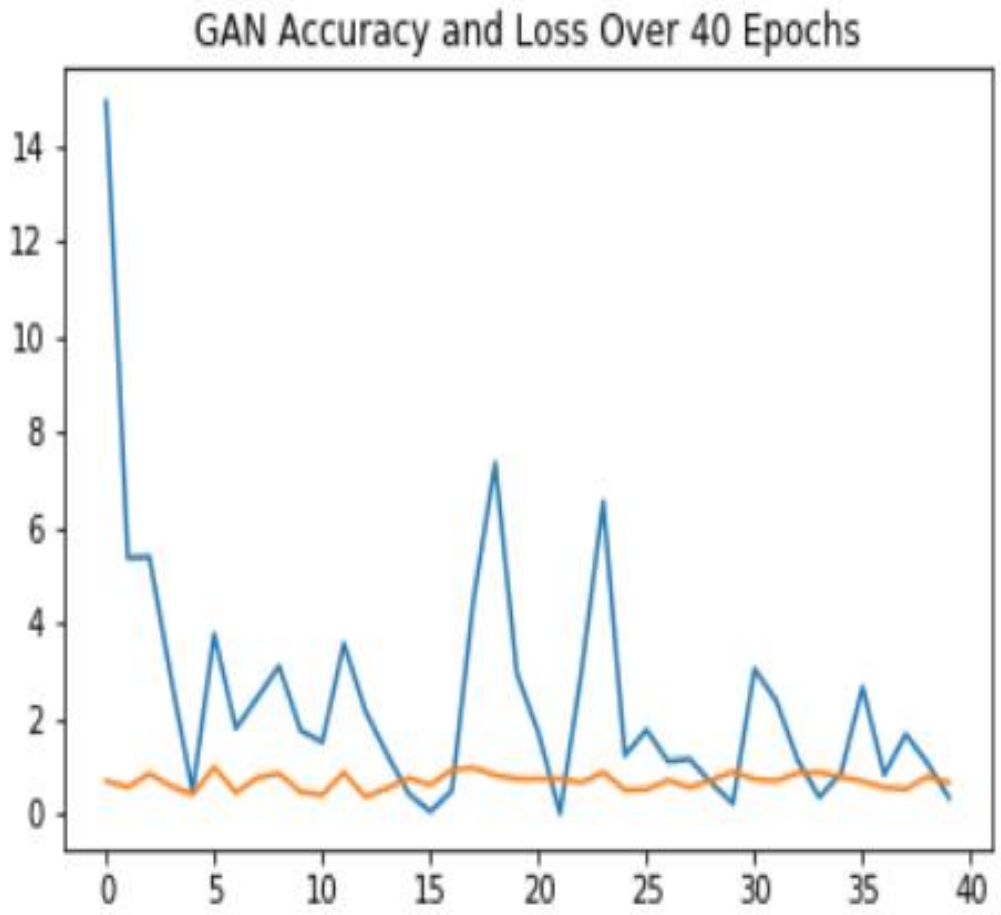


Figure 5.4 Graph between GAN accuracy and *gen1_model* loss

Figure 5.4 shows the accuracy versus loss graph for the GAN model trained with images in LAB color space. As the number of epochs goes on increasing the overall generator loss goes on decreasing. The accuracy of the model is almost near about 1 and sometimes it is exactly 1.

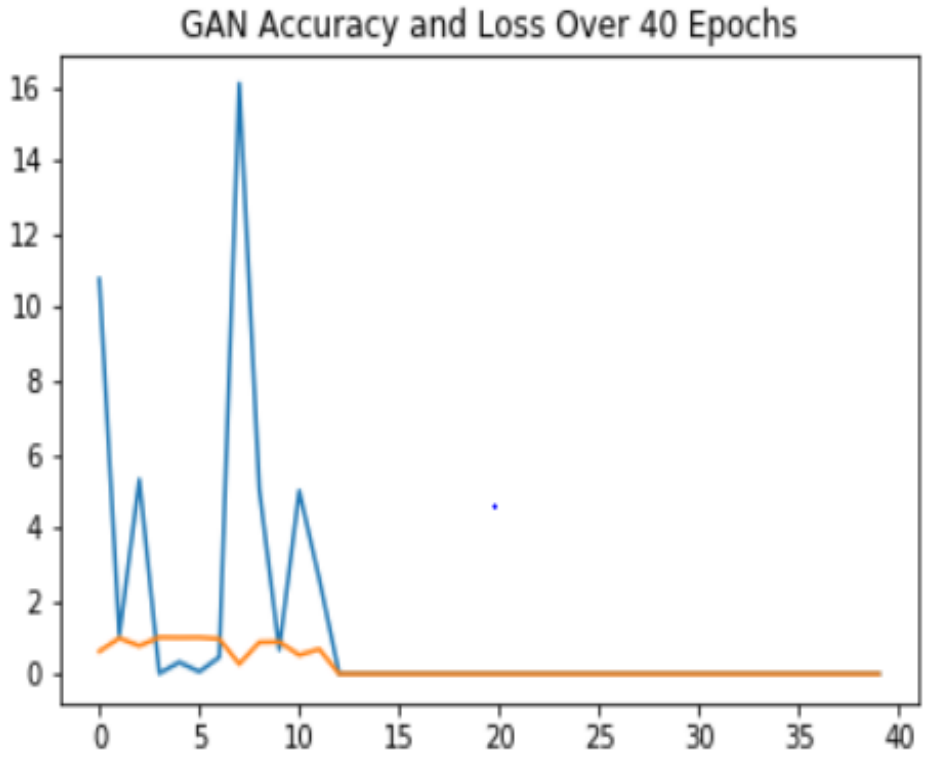


Figure 5.5 Graph between GAN accuracy and *gen2_model* loss

Figure 5.5 shows the accuracy versus generator loss for the generator model trained with YCbCr images. In this graph, first the generator loss decreases but then it reaches to its peak at 6-7 epoch and then gets decreases and becomes zero after 11 or 12 epochs. At the same time, the accuracy provided by this model is not so good. So the images generated using this color space don't have that much good quality what an image with LAB color space has.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

CIE-LAB color space represents the color accurately than any other color space. It is device-independent. A CIE-LAB color space has a wide range of colors than any other color space. From the experimental results, it is concluded that CIE-LAB color space is better than the YCbCr color space. During training, it is observed that introducing a dropout layer in the model reduces the model losses up to a significant value.

6.2 FUTURE WORK

Colorization of grayscale images is useful in the photography industry as well in the medical field. In the medical field, these GANs are also used to detect some particular diseases. So in our work, we have performed colorization of whole image. In the future, this GAN model can be further modified to color the selected parts of an image. It can be used for the head lighting purpose. For example, if in the image of a breast there are some cells that have the symptoms of cancer, first they are detected using discriminator model and then highlighted using the colorization technique. This makes easy to differentiate between damaged and undamaged cells.

REFERENCES

[1] Visual-reverse-image-search. https://image.shutterstock.com/image-photo/beautiful-water-drop-on-dandelion-260nw-789676552.jpg

[2] V. Bochko, P. V¨alisuo, T. M. Alho, S. Sutinen, J. Parkkinen, and J. T. Alan-der, "Medical image colorization using learning," in Conference on Colourin Graphics, Imaging, and Vision, vol. 2010, no. 1. Society for Imaging Science and Technology, 2010, pp. 70–74.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Ward-Farley,S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing systems, 2014, pp. 2672–2680.

[4] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," arXiv preprint arXiv:1511.08458, 2015.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in Proceedings of the IEEE Conference on computer vision and pattern recognition, 2017, pp. 1125–1134.

[6] D. Ciresan, U. Meier and J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification," arXiv preprint arXiv:1202.2745, 2012.

[7] Convolutional Neural Network. https://miro.medium.com/max/1838/1*uAeANQIoQPqWZn - uHVEyw.jpeg

[8] D. Scherer, A. Muller and S. Behnke, "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition," in 20th International Conference on Artificial Neural Networks, 2010, pp. 92-101.

[9] Y. Bengio, E. T. Laufer, G. Alain and J. Yosinski, "Deep Generative Stochastic Networks Trainable by Backprop," in Proceedings of the 31st International Conference on Machine Learning, 2014, pp. 226-234.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," in Journal of Machine Learning Research 15, 2014, pp. 1929-1958.

[11] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in Proceedings of the 32nd International Conference on Machine Learning, arXiv preprint arXiv:1502.03167, 2015.

[12] RGB color space. https:// www.baslerweb.com/fp1485687434/media/editorial/content_images/faqs/faq_RGB_1.gif

[13] CIE-LAB color space. https://colorapplications.com/wp-content/uploads/2018/09/3D-LAB-300x300.jpg

[14] N. Fdhal, M. Kyan, D. Androutsos and A. Sharma, “Color Space Transformation from RGB to CIELAB Using Neural Networks,” in Advance in Multimedia Information Processing, 2009, pp. 1011-1017.

[15] A. Sharma, “Understanding Color Management,” John Wiley & Sons, 2018.

[16] P. Kakumanu, S. Makrogiannis and N. Bourbakis, “A survey of skin-color modeling and detection methods,” in Journal of Pattern Recognition, 2007, vol 40, pp. 1106 – 1122.

[17] YCbCr color space. https://scc.ustc.edu.cn/zlsc/sugon/intel/ipp/ipp_manual/IPPI/ippi_ch6-/Images/ch6_color_models_4.jpg

[18] Generative adversarial network. https://miro.medium.com/max/1200/1*pHOkZ0HJrUSP827-fZFETg.png

[19] Conditional GAN. https://miro.medium.com/max/960/1*-buiUh_PpX_XkNOzwt83Cw.png

[20] R. Zhang, P. Isola and A. A. Efros, “Colorful Image Colorization,” in European Conference on Computer Vision, 2016, pp. 649-666.

[21] A. Levin, D. Lischinski and Y. Weiss, “Colorization using Optimization,” in Conference of ACM transactions on graphics, 2004, vol 23, pp. 689-694.

[22] A. Radford, L. Metz and S. Chintala, “Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks,” arXiv preprint arXiv:1511.06434, 2016.

[23] T. Welsh, M. Ashikhmin and K. Mueller, “Transferring Color to Greyscale Images,” in Conference of ACM transactions on graphics, 2002, vol 21, pp. 277-280.

[24] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” arXiv preprint arXiv:1411.1784, 2014.

[25] S. Koo, “Automatic Colorization with Deep Convolutional Generative Adversarial Networks,” CS231n in Stanford University, 2016.

[26] J. Hwang and Y. Zhou, “Image Colorization with Deep Convolutional Neural Networks,” CS231n in Stanford University, 2016.

[27] Y. Cao, Z. Zhou, W. Zhang and Y. Yu, “Unsupervised Diverse Colorization via Generative Adversarial Networks,” in Joint European Conference on Machine Learning, 2017.

[28] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,*”* in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, 2010, pp. 249-256.

[29] R. k. Gupta, A.Y.-S. Chia, D. Rajan, E.S. Ng and H. Zhiyong, “Image Colorization Using Similar Images,” in Proceedings of the 20th ACM international conference on Multimedia, 2012, pp. 369-378.

[30] V. Jain and H. S. Seung, “Natural Image Denoising with Convolutional Networks,” in Advances in neural information processing systems, 2009, pp. 769-776.

[31] B. Mazoure, T. Doan and S. Ray, “EmojiGAN: learning emojis distributions with a generative model,” in Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, 2018, pp. 273-279.

[32] C. Ledig, L. Theis, F. Huszár and J. Caballero, “Photo- Realistic Single Image Super-Resolution Using a Generative Adversarial Network,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4681-4690.

[33] P. L. Suarez, A. D. Sappa and B. X. Vintimilla, “Infrared Image Colorization based on a Triplet DCGAN Architecture,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 212-217.

[34] A. Jaiswal, W. AbdAlmageed, Y. Wu and P. Natarajan, “Bidirectional Conditional Generative Adversarial Networks,” in Asian Conference on Computer Vision, 2018, pp. 216-232.

[35] O. Ronneberger, P. Fischer and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in International Conference on Medical image computing and computer-assisted intervention, 2015, pp. 234-241.

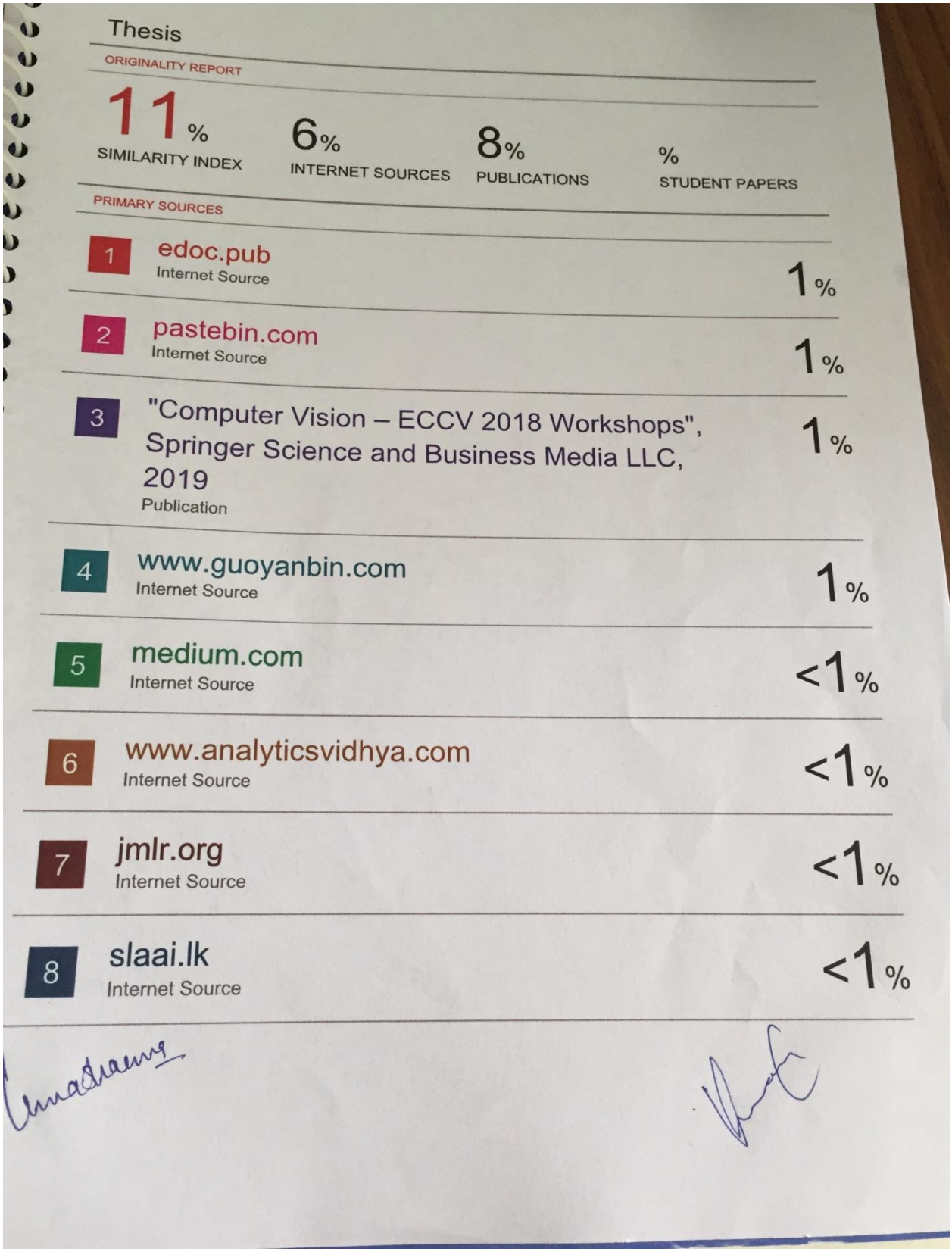
[36] X. Yi, E. Walia and P. Babyn, “Generative Adversarial Network in Medical Imaging: A Review,” arXiv preprint arXiv:1809.07294, 2018.

[37] H. Zenati, C. S. Foo, B. Lecouat, G. Manek and V. R. Chandrasekhar, “Efficent GAN based Anomaly Detection,” arXiv preprint arXiv:1802.06222, 2018.

[38] K. Armanious, C. Jiang, M. Fischer, T. Kustner, K. Nikolaou, S. Gatidis and B. Yang, “MedGAN: Medical Image Translation using GANs,” arXiv preprint arXiv:1806.06397, 2018.

[39] Q. Yang, P. Yan, Y. Zhang, H. Yu, Y. Shi, X. Mou, M. K. Kalra, Y. Zhang, L. Sun and G. Wang, “Low Dose CT Image Denoising Using a Generative Adversarial Network with Wasserstein Distance and Perceptual Loss,” in IEEE transactions on medical imaging, 2018, vol 37, pp. 1348-1357.

[40] D. C. Ciresan, A. Giusti, L. M. Gambardella and J. Schmidhuber, “Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks,” in International Conference on Medical Image Computing and Computer-assisted Intervention, 2013, pp. 411-418.



Thesis

ORIGINALITY REPORT

11%	6%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	edoc.pub Internet Source	1%
2	pastebin.com Internet Source	1%
3	"Computer Vision – ECCV 2018 Workshops", Springer Science and Business Media LLC, 2019 Publication	1%
4	www.guoyanbin.com Internet Source	1%
5	medium.com Internet Source	<1%
6	www.analyticsvidhya.com Internet Source	<1%
7	jmlr.org Internet Source	<1%
8	slaai.lk Internet Source	<1%