A

Seminar Report

On

# Image Colorization using Deep Learning

Submitted to the Department of Electronics Engineering in Partial Fulfilment for the Requirements for the Degree of

**Bachelor of Technology**

**(Electronics and Communication)**

by

**Mr. Shubh Kawa**

**(U21EC011)**

**(B. TECH. III(EC), 5$^{th}$ Semester)**

Guided by

**Dr. Kishor Upla**

**Assistant Professor, ECED**

DEPARTMENT OF ELECTRONICS ENGINEERING

SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY

NOV23

**Sardar Vallabhbhai National Institute Of Technology**

Surat - 395 007, Gujarat, India

DEPARTMENT OF ELECTRONICS ENGINEERING

# CERTIFICATE

This is to certify that the SEMINAR REPORT entitled "**Image Colorization using Deep Learning**" is presented & submitted by Candidate Mr. Shubh Kawa, bearing Roll No.U21EC011, of B.Tech. III, 5th Semester in the partial fulfillment of the requirement for the award of B.Tech. Degree in Electronics & Communication Engineering for academic year 2023 - 24.

He/She has successfully and satisfactorily completed his/her Seminar Exam in all respects. We certify that the work is comprehensive, complete and fit for evaluation.

**Dr. Kishor Upla**
Assistant Professor & Seminar Guide

| **Name of Examiners** | **Signature with Date** |
|---|---|
| 1.Dr. P.K. Shah | ____________________ |
| 2. Dr. K. Captain | ____________________ |

**Dr. J.N. Sarvaiya**
Head & Associate Professor
DECE, SVNIT

Seal of The Department
(November 2023)

# Acknowledgements

I would like to express my profound gratitude and deep regards to my guide Dr. Kishor Upla for his guidance. I am heartily thankful for suggestion and the clarity of the concepts of the topic that helped me a lot for this work. I would also like to thank the head of the Electronics Engineering Department, SVNIT and all the faculties of ECED for their cooperation and suggestions. I am very much grateful to all my classmates for their support.

Shubh Dinesh Kawa
Sardar Vallabhbhai National Institute of Technology
Surat

Nov 4, 2023

# Abstract

Image colorization is a fundamental task in computer vision that involves the process of adding colors to grayscale images. This task has witnessed significant advancements with the advancement of deep learning techniques.

The report surveys deep learning architecture(Convolutional Neural Networks) and methodologies employed in image colorization which include autoencoders, and Generative Adversarial Networks. An implementation is performed that harnesses the power of transfer learning using the pre-trained model of VGG-16 and using a custom detector to output different channels.

The custom datasets created are explained, with the accuracy of each dataset shown. Satisfactory results are obtained considering the dataset and the model parameters, which are also shown.

This report concludes with the summary of the complete report, the insights formed and the future scopes of the current implementation

Name: Shubh Kawa
Admission No. : U21EC011
Guide Name: Dr. K.P. Upla
Seminar Date: 4th November, 2023

Examiner's Name
Dr. Prashant.K Shah
Dr. Kamal Captain

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Image Colourization is the process of converting grayscale images to colorized images. Earlier this process used to be extremely tedious as it was done manually, after the introduction of Deep Learning, this process is being automated. The complexity of colorization arises from the necessity to handle a variety of imaging situations with a single method. The issue is also gravely ill-posed because two of the three image dimensions are lacking, even though scene semantics, such as the fact that grass is typically green, clouds are typically white, and the sky is typically blue, may be useful in many situations. However, for many man-made and natural objects, like shirts, cars, and flowers, such semantic priors are incredibly rare. Additionally, the colorization problem inherits the common difficulties associated with picture improvement, including changes in lighting, differences in views, and occlusions.

Image colorization even though seems like a problem that doesn't need to be solved, has many significances. Photographs of old historical pictures which were present in black and white can be made colorful to add a more immersive experience for the viewers. It has been seen that colorful images has an enhanced visual understanding as colorized images provide a clearer understanding of the objects depicted.

Even though Deep Learning has provided many solutions the accuracy is quite difficult. This problem has been researched by many individuals and many have provided efficient solutions. Deep Learning has had a profound effect on today's world, we can observe that Artificial Intelligence is taking over almost every industry, and Deep Learning in particular.

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. Deep learning is preferred over machine learning as it provides more efficient results and in deep learning the problem is solved end to end whereas in machine learning, we have to break the problem statement into parts to get the solution. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. The main advantage of Deep Learning is it thinks like a human brain which in turn gives us splendid results. Research in deep learning is very active, leading to continual improvements in model architectures, training techniques, and performance across various tasks. The figure 1.1 shown below gives an architecture of how neural networks look.

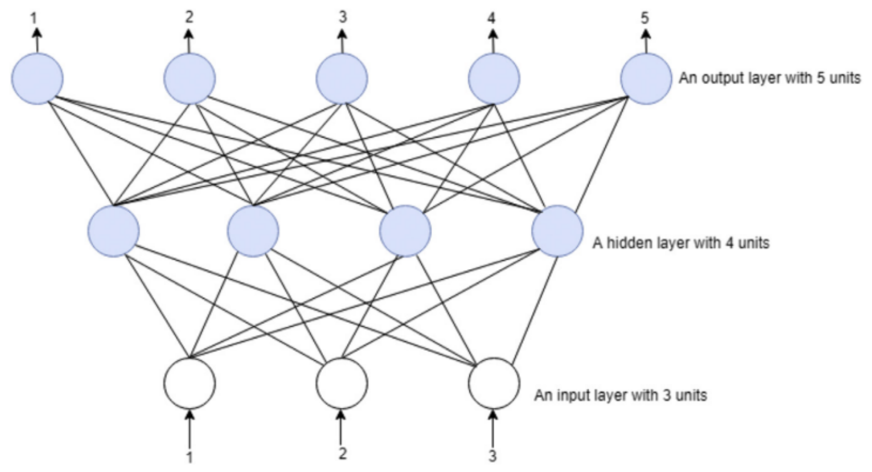

1
2
3
4
5
An output layer with 5 units
A hidden layer with 4 units
An input layer with 3 units
1
2
3

Figure 1.1: A 2 layer neural network structure [1]

## 1.1 Motivation

The motivation that made me choose the topic was my keen interest in Deep Learning and its applications. The fact that we use Artificial Intelligence almost every day and we do not even realize it blows my mind. Deep Learning is providing solutions to better humanity, every noble thing has a dark side and our goal as responsible citizens is to focus on and contribute to the noble side. Every image holds a memory for us, it reminds us of the little packets of happiness that we treasure in this life. Just imagine a colorless life, it would be so dull, meaningless, and naturally depressing. The sunrise we see when we wake up, those vivid colors give us hope that our life has meaning, and the sunset indicates that darkness is inevitable. Colors hold an extremely important significance.

My main motivation in pursuing this project is for my grandparents. Recently my grandparents were showing me their photo album and it was extremely beautiful, however, my grandmother expressed how she wished all those images were colored. This made me realize that I can harness deep learning's power to accomplish this.

## 1.2 Objectives

I aim to attempt to solve this problem using transfer learning and by creating a custom detector. I aim to understand what conditional generative adversarial networks are, how they work, and how I can implement them using the different datasets. I aim to have a thorough understanding of the CIELAB colorspace and how it will be advantageous to us.

## 1.3 Flow of Report

After going through several solutions, there was a common term *autoencoders*. Almost all autoencoders used in Image Colorization use Convolutional Neural Networks. We start by understanding what CNNs are, what types of layers they comprise, and what uses they have. We then jump onto autoencoders to understand how specifically are they helpful in Image Colorization. For implementation, I harness the power of transfer learning using the VGG-16 model and for classification, a custom CNN detector is created. In the recent years, Conditional Generative Adversarial Networks have been used in image colorization. So we talk about Conditional GANs and how they help us to get better output in Image Colorization. At last, I showcase my results with reference to the dataset created.

# Chapter 2
# Literature Review

After going through several research papers to help me solve this problem, there were several keen observations I noted, the following sections elaborates those observations

## 2.1 Recent Work

Image Colorization is a problem that many researchers are working on as it is challenging to correctly identify which color is to be segmented in that particular area. Saed Anwar and others [12] contributed by providing a thorough review of several image colorization techniques. They observed the following trends, GAN based methods deliver diverse colorization visually compared to CNN-based methods, the existing models generally deliver a sub-optimal result for complex scenes having a large number of objects with small sizes, They further concluded that the future direction of image colorization is unsupervised learning, and many recent advancements and techniques such as attention mechanism, and loss functions can be incorporated for performance.

There are several methods to train a model, our goal is to always reduce the training time and increase our efficiency. By reading the paper by Jeff Hwang and others [13], they built a learning pipeline that comprises a neural network and an image pre-processing front-end. It helped me develop a sense of intuition about how to take advantage of transfer learning of pre-trained models and how to tune it for my purpose. A key idea of their model was that they used a concatenation layer, which was inspired by segmentation methods. Combining multiple intermediate feature maps in this fashion has been shown to increase the prediction quality in segmentation problems, producing finer details and cleaner edges. Their results showed that the ADAM optimizer had a faster convergence. They placed a batch normalization layer before every non-linearity layer apart from the last few layers before the output. In their trials, they have found that doing so does improve the training rate of the systems. They conducted multiple trials of training with minibatch updates to see which learning rate produced faster convergence behavior over a fixed number of iterations in order to determine an appropriate learning rate. They discovered that a learning rate of 0.001 was present in the set of learning rates sampled on a logarithmic scale, attaining both the lowest training loss of the learning process and one of the biggest per-iteration decreases in training loss sampled rates. With that as a foundation, they proceeded to utilize the full training package. Using a hold-out percentage of 10% as the validation set, they found that convergence was the fastest with a 0.0003 learning rate.

An Activation Function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations. The role of the Activation Function is to derive output from a set of input values fed to a node. In broad terms, activation functions are necessary to prevent linearity. Without them, the data would pass through the nodes and layers of the network only going through linear functions (a*x+b). The composition of these linear functions is again a linear function and so no matter how many layers the data goes through, the output is always the result of a linear function. In the work of Singh [14], they used the tanh activation function which significantly boosted their results.The figure 2.1 shows how the activation function works.

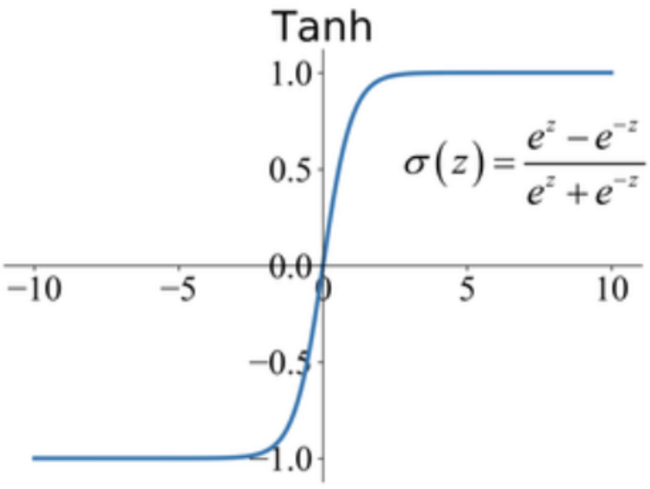


Figure 2.1: Tanh activation function [2]

In recent years, there has been a blast of neural networks in the market and Generative Adversarial Networks are one of the, it has shown significant development in the field of image colorization. The training procedure functions as a game of skill: the discriminator strives to improve at telling real from false, while the generator attempts to produce data that is identical to real data. This adversarial process has the potential to provide very convincing synthetic data over time for the generator. Finding the ideal balance during training is one of the main issues with GANs. If the discriminator is too strong, the generator may not learn enough, or it may begin to generate data that is overfitting—that is, data that is too similar to the real data. Since their introduction by Ian Goodfellow and associates in 2014, GANs have emerged as a hot topic for AI and machine learning research. They have brought about a great deal of noteworthy progress in generative modeling. After reviewing the work of Kamyar Nazeri and others [15], a Generative Adversarial Network is very effective in image colorization, they compared their GAN model with their pre-existing CNN model. Their method was as follows A high dimensional input is mapped to a high dimensional output in the picture colorization problem, which is an image-to-image translation problem. It can be viewed as a pixel-by-pixel regression problem when the input and output structures are very closely aligned. This means that each pixel in the grayscale input image must get color information in addition to being output in the same spatial dimension as the input by the network. Using a regression loss as our starting point, they presented a complete convolutional model architecture and subsequently expanded the concept to adversarial nets.

For training their network, they utilized the Adam [16] optimizer. Every model parameter has its own learning rate that ADAM keeps track of. During training, it modifies these rates dynamically according to the estimated first and second moments of the gradients.ADAM speeds up the optimization process by utilizing the momentum idea. An exponentially decaying moving average of previous gradients is monitored.Additionally, a moving average of previous squared gradients that is exponentially declining is monitored by ADAM. This aids in scaling the learning rates for various parameters in an adaptive manner. These two methods are combined by the algorithm to adaptively modify the momentum and learning rates for every parameter. This implies that it can deal with scenarios in which distinct model components may have different optimal learning rates. They would manually decay the learning rate by a factor of 10 whenever their loss function would start to get constant, this was a very interesting aspect. Instead of using the traditional *ReLU* function, Ratford et al. [17] showed that training the discriminator with *LeakyReLU* activation function gives better performance. They utilized the L*a*b color space for the colorization task. They follow this color space because it contains dedicated channels to depict the brightness of the image and the color information is fully encoded in the remaining two channels.

## 2.2 Traditional Work

To add color to grayscale photographs, a procedure known as image colorization was started by researchers in the early 2000s. This project was a major turning point in the field of computer vision research. Welsh et al. [18] in 2002, presented a groundbreaking approach that uses texture synthesis to colorize images. Their method required comparing a target grayscale image with a corresponding colored reference image in terms of texture and brightness information. They were able to determine and add the proper colors to the grayscale image by aligning these features.
Levin et al. [19] also suggested a different approach to image colorization. They approached the problem from the opposite direction, creating a cost function that punished the difference between each grayscale image pixel and the weighted average of its sur-

rounding pixels.
Notwithstanding the encouraging developments, these early methods had a serious flaw. Their practical application was limited since the colorization process required significant human interaction. Therefore, more research and development were needed to provide automatic and effective methods for image colorization.

# Chapter 3
# Convolutional Neural Networks

In the field of Computer Vision, we can establish that convolutional neural networks are the base.

## 3.1 What are convolutional Neural Networks

Before diving into CNNs, let's first understand what are neural networks.

Neural networks are a class of machine learning models inspired by the structure of the human brain. They are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling, or clustering of raw input. They are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

After getting a fundamental understanding of Neural Networks, let us understand about Convolutional Neural Networks

One subset of machine learning is the convolutional neural network, sometimes known as a convnet or CNN. It is one of the many varieties of artificial neural networks that are employed for diverse kinds of data and applications. For deep learning techniques, a CNN is a type of network design that is particularly useful for applications involving the processing of pixel input, such as image recognition.

CNNs are the preferred network architecture for object identification and recognition in deep learning, while there are other varieties of neural networks as well. They are therefore ideal for computer vision (CV) jobs and critical object recognition applications like facial recognition and self-driving cars. The design of a CNN is comparable to the structure of connectivity found in the human brain. Similar to how the brain is made up of billions of neurons, each neuron in a CNN is structured differently. The arrangement of neurons in a CNN is actually similar to that of the frontal lobe of the brain, which processes visual stimuli. Because the full visual field is covered by this configuration, classic neural networks do not face the issue of piecemeal image processing, where images must be fed in reduced resolution segments. A CNN performs better when it comes to image inputs, speech or audio signal inputs, and other input types than the previous networks.

## 3.2 Layers present in Convolutional Neural Networks

- **Input layers:-** This is always the first layer in our model. Generally, the input will be an image or a sequence of images.
- **Convolutional Layers:-** We use this layer to extract features from our input. We apply a set of learnable filters known as kernels to the input images. Kernels are usually smaller matrices of the size 2*2, 3*3, or 5*5 in shape. It slides over the input and computes the dot product between the kernel weight and the corresponding input. The output of this layer is known as feature maps.
- **Activation Layer:-** Activation layers give the network nonlinearity by appending an activation function to the output of the layer that came before it. It will take the output of the convolution layer and apply an element-wise activation function. Activation functions like Tanh, Leaky RELU, max(0, x), and RELU are frequently used.
- **Pooling Layer:-** This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. In max pooling , the maximum value present in that filter is taken and in average, the average of all the values is taken. Below figure 3.1 depicts the architecture of CNN.

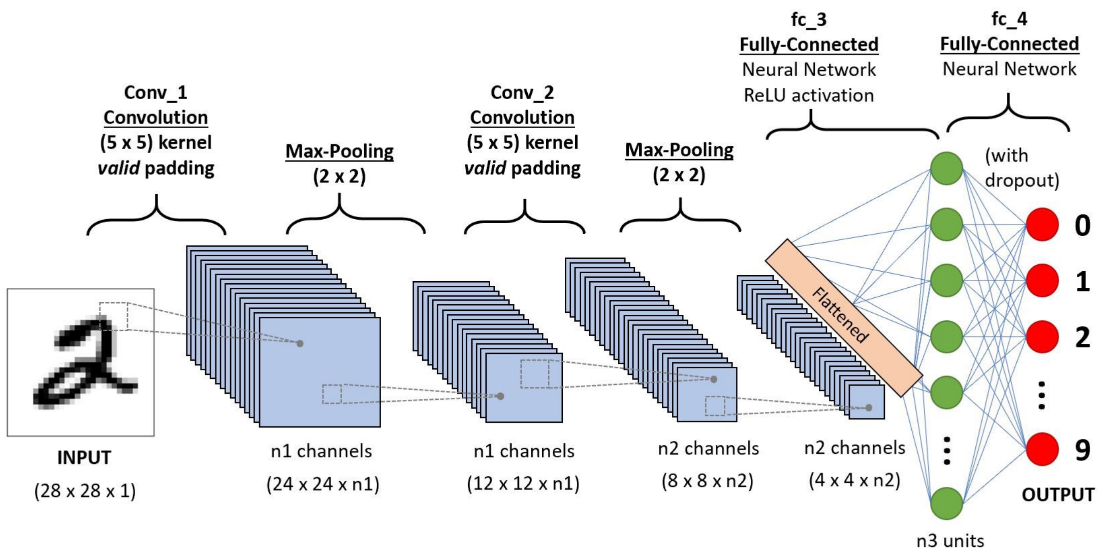


Figure 3.1: Architecture [3]

- **Flatten Layer:-** The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.
- **Fully Connected Layer:-** It takes the input from the previous layer and computes the final classification or regression task.
- **Output Layer:-** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

## 3.3 How do Convolutional Neural Networks work?

A normal image has the following attributes, height, width and the RGB contents spread over three channels which are R, G and B respectively.
An image 3.2 can be visualized as shown in

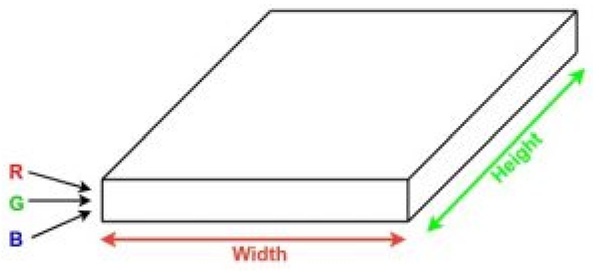


Figure 3.2: Image Representation [4]

Convolutional Layers work in such a way that a small patch of this image runs a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called Convolution. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights. Below figure5.1 shows how convolution works

### Convolutional Process

- A collection of learnable filters, or kernels, with modest widths, heights, and the same depth as the input volume make up convolution layers (3 if the input layer is image input).

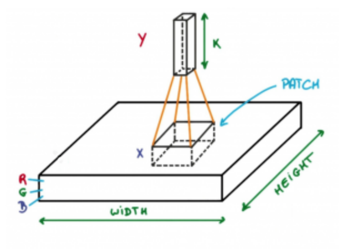


Figure 3.3: Convolutional layers working [4]

- For example, if we have to run convolution on an image with dimensions 34x34x3. The possible size of filters can be axax3, where 'a' can be anything like 3, 5, or 7 but smaller as compared to the image dimension.
- In the forward pass, we calculate the dot product between the kernel weights and patch from input volume and slide each filter across the entire input volume step by step. Each step is called a stride, and it can be as small as 2, 3, or even 4 for high-dimensional pictures.
- Each filter we slide will provide a 2-D output, which we will stack together to produce an output volume with a depth equal to the number of filters. Every filter will be learned by the network.

**Most Frequent Used Activations**

- **Sigmoid:-** The function of the sigmoid is nonlinear. It converts a real-valued number into a range of values between 0 and 1.

  Nonetheless, a highly undesirable characteristic of the sigmoid is that the gradient nearly vanishes when the activation is at either tail. In backpropagation, an extremely small local gradient will essentially "kill" the gradient. Additionally, if the neuron receives only positive input, the sigmoid's output will contain either positive or negative values, creating a zigzag pattern in the gradient updates for weight.
- **Tanh:-** Tanh squashes a real-valued number to the range [-1, 1]. Like sigmoid, the activation saturates, but — unlike the sigmoid neurons — its output is zero centered.
- **ReLU:-** In recent years, the Rectified Linear Unit (ReLU) has gained a lot of popularity. The function F(k)=max (0,k) is computed. Stated otherwise, the activation is only the threshold at zero. ReLU is more dependable than sigmoid and tanh, and it speeds up convergence by six times. Regretfully, one drawback of ReLU is that it might be brittle during training. It can be updated in such a way

that it never receives another update from a huge gradient passing past it. We can deal with this, though, if we establish an appropriate learning rate.

## 3.4 Motivation behind convolution

Convolution makes use of sparse interaction, parameter sharing, and equivariant representation—three key concepts that inspired computer vision researchers. Let's go into more depth about each of them.

Matrix multiplication is used by trivial neural network layers to describe the interaction between the input and output unit through a matrix of parameters. This implies that all output units communicate with all input units. Convolution neural networks, on the other hand, exhibit sparse interaction. This is accomplished by shrinking the kernel relative to the input. For example, even though an image may include millions or thousands of pixels, significant information can be found within tens or hundreds of pixels when processing the image with a kernel. This implies that storing fewer parameters will lower the model's memory requirements while simultaneously increasing its statistical efficacy.

If a feature at a spatial position (x1, y1) can be computed, it should be possible to compute the same feature at another spatial point, say (x2, y2). It indicates that neurons are limited to using the same set of weights for a single two-dimensional slice, or for generating a single activation map. In a convolution network, weights supplied to one input are applied to other inputs in order to obtain the same output, whereas, in a typical neural network, each member of the weight matrix is utilized once and never again.

The convolution neural network's layers will be equivariant to translation because of parameter sharing. It states that any changes we make to the input will correspondingly affect the outcome.

## 3.5 Applications of CNNs

- **Object Detection:** Thanks to CNN, we now have advanced models such as R-CNN, Fast R-CNN, and Faster R-CNN, which are widely used as the pipeline for many object detection models used in facial recognition, autonomous vehicles, and other applications.

- **Semantic Segmentation:** A team of Hong Kong academics created the CNN-based Deep Parsing Network in 2015 in order to integrate rich data into an image segmentation model. Additionally, UC Berkeley researchers developed fully convolutional networks that outperformed the most advanced semantic segmentation techniques.

- **Image Captioning:** Recurrent neural networks are employed alongside CNNs to provide captions for pictures and videos. Numerous uses for this exist, like activity recognition and providing the visually handicapped with descriptions of movies and visuals. YouTube has significantly utilized it in order to make sense of the enormous quantity of videos that are regularly posted to the network.

# Chapter 4
# Autoencoders

In this section, We will talk about autoencoders, one of the most fundamental networks that has an application in image colorizations.

## 4.1 What are Autoencoders?

Autoencoder is a type of neural network. The output layer of an autoencoder has the same dimensionality as the input layer. Put another way, there are exactly as many output units in the output layer as there are input units in the input layer. An autoencoder, also known as a replicator neural network, is a type of neural network that duplicates data unsupervised from the input to the output.

By putting the input through the network, the autoencoders recreate each dimension. While using a neural network to replicate an input may seem simple, the input is reduced in size into a smaller representation during the replication process. The middle layers of the neural network have a fewer number of units as compared to the input or output layers. Therefore, the middle layers hold a reduced representation of the input. The output is reconstructed from this reduced representation of the input.

## 4.2 Autoencoder Architecture

Autoencoders consists of three components as shown in the figure 4.1

#### Encoder

An encoder is a feedforward, fully connected neural network that compresses the input into a latent space representation and encodes the input image as a compressed representation in a reduced dimension. The compressed image is the distorted version of the original image. The encoder typically consists of multiple layers composed of neurons (nodes), with each layer gradually reducing the dimensions.

#### Latent Space

This constitutes the part that contains all the crucial information about the data given to the input layer. It is a reduced representation of the input fed into the decoder.

### Decoder

This is the second part of the autoencoder. It takes the lower-dimensional representation (encoder output) and attempts to reconstruct the original input data. Like the encoder, the decoder consists of multiple layers in reverse order. The final layer of the decoder produces the reconstructed output.

All of the above layers comprise the autoencoder architecture which is shown in the figure 4.1. The architecture gives us an intuition of how autoencoders work and their general structure.

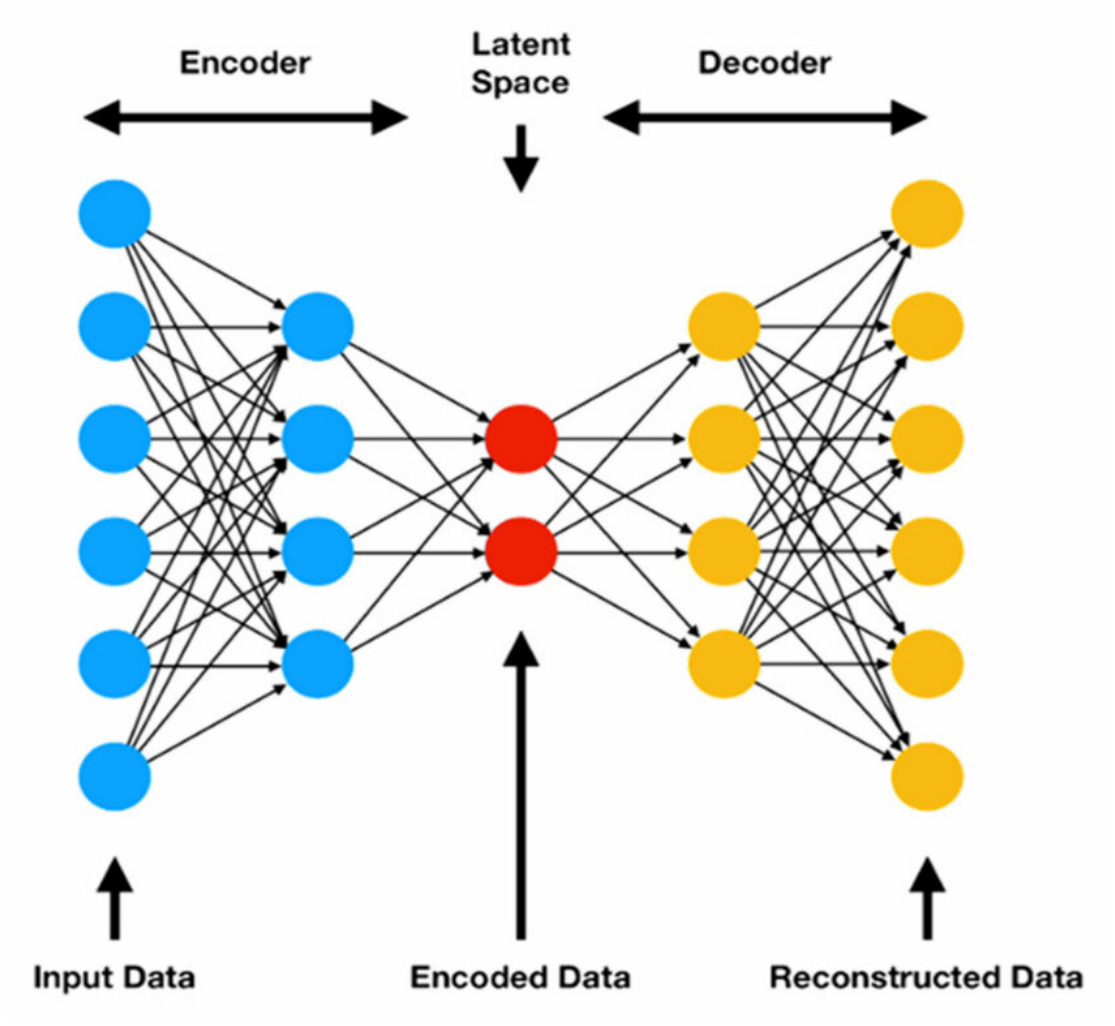


Figure 4.1: Autoencoder Architecture [5]

## 4.3 Types of Autoencoders

### Convolutional Autoencoder

Convolutional Autoencoders (CAE) derive the input's encoding from a collection of basic signals, which they subsequently use to reconstruct the original signal. Moreover, we can use CAE to create the image's reflectance or change its geometry. The encoder layers in this kind of autoencoder are called convolution layers, while the decoder levels are called deconvolution layers. Upsampling and transpose convolution are other names for the deconvolution side.

### Advantages

1. Due to their convolutional nature, they scale well to realistic-sized high-dimensional images
2. They can help in removing noise or reconstructing missing parts

### Drawbacks

The reconstruction of the input image is often blurry and of lower quality due to compression during which information is lost.

### Variational Autoencoders

Variational autoencoder models make strong assumptions concerning the distribution of latent variables. They employ a variational method for latent representation learning, which yields an extra loss component and a particular estimate known as the Stochastic Gradient Variational Bayes estimator for the training process. It is assumed that a directed graphical model created the data and that the encoder is learning an approximation to the posterior distribution, where theta and  stand for the parameters of the respective encoder (generative model) and decoder (recognition model). Compared to a regular autoencoder, the probability distribution of the latent vector in a variational autoencoder usually closely resembles that of the training data.

### Advantages

1. It gives significant control over how we want our model to produce latent distribution, this isn't present in other types of autoencoders
2. After training, we can sample from the distribution followed by decoding and generating new data

### Drawbacks

When training a model, there is a need to calculate the relationship of each parameter in the network concerning the final output loss using a technique known as backpropagation. Due to this, the sampling process requires extra attention. Undercomplete Autoencoder    The objective of the under complete autoencoder is to capture the most important features present in the data. Undercomplete autoencoders have a smaller dimension for the hidden layer compared to the input layer. This helps to obtain important features from the data. It minimizes the loss function by penalizing the g(f(x)) for being different from the input x.

### Advantages

Under complete autoencoders do not need any regularization as they maximize the probability of data rather than copying the input to the output

### Drawbacks

Using an overparameterized model due to a lack of sufficient training data can create overfitting.

### Denoising Autoencoder

Denoising autoencoders creates a corrupted copy of the input by introducing some noise. This helps to prevent the autoencoders from copying the input to the output without learning features about the data. These autoencoders take a partially corrupted input while training to recover the original undistorted input. The model learns a vector field for mapping the input data towards a lower dimensional manifold which describes the natural data to cancel out the added noise.Below figure4.2 shows the working of denoising autoencoder.

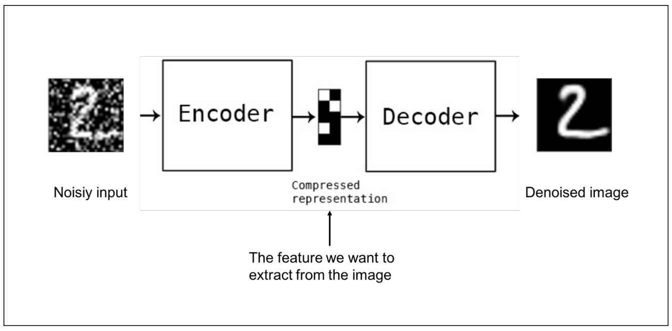

Encoder
Decoder
Noisiy input
Compressed representation
Denoised image
The feature we want to extract from the image

Figure 4.2: Example of denoising autoencoder [6]

**Advantages**

1. Corruption of the input can be done randomly by making some of the inputs zero. The remaining nodes copy the input to the noised input
2. It is quite easy to set up a single threat denoising autoencoder

**Drawbacks**

The model is not able to develop a mapping that memorizes the training data because our input and target output are no longer the same

**Sparse Autoencoder**

Sparse autoencoders have hidden nodes greater than input nodes. They can still discover important features from the data. A generic sparse autoencoder is visualized where the obscurity of a node corresponds with the level of activation. Sparsity constraint is introduced on the hidden layer. This is to prevent the output layer from copying input data.

**Advantages**

They take the highest activation values in the hidden layer and zero out the rest of the hidden nodes. This prevents autoencoders from using all of the hidden nodes at a time and forces only a reduced number of hidden nodes to be used.

**Drawbacks**

For it to be working, it's essential that the individual nodes of a trained model that activate are data dependent, and that different inputs will result in activations of different nodes through the network.

**Deep Autoencoder**

Deep Autoencoders consist of two identical deep belief networks, one network for encoding and another for decoding. Typically deep autoencoders have 4 to 5 layers for encoding and the next 4 to 5 layers for decoding. We use unsupervised layer-by-layer pre-training for this model. The layers are Restricted Boltzmann Machines which are the building blocks of deep-belief networks.

**Advantages**

Final encoding layer is compact and fast

**Drawbacks**

Chances of overfitting occur as there are more parameters than input data.

## 4.4 Applications of Autoencoders

- **Anomaly Detection:-** Autoencoders can learn to reconstruct normal data, so when presented with anomalous data, they produce higher reconstruction errors, making them useful for anomaly detection
- **Image Colorization:-** By training an autoencoder on grayscale images and using the encoder-decoder architecture, one can effectively colorize images.
- **Recommendation Systems:-** Autoencoders can be used to learn compressed representations of user-item interaction data in recommendation systems.
- **Generative Modelling:-** Variational Autoencoders (VAEs) can generate new data points that are similar to those in the training set.

# Chapter 5
# Implementation Using Transfer Learning

After thoroughly understanding autoencoders and CNNs, the next step was implementing the model. Now due to being constrained by computationally power, I decided to harness the power of Tranfer Learning.

## 5.1 Transfer Learning

Transfer learning, used in machine learning, is the reuse of a pre-trained model on a new problem.A machine uses transfer learning to increase its generalization about another task by using the knowledge it has learned from a prior one. For instance, you may use the knowledge a classifier learned to identify drinks while training it to predict whether an image contains food.Transfer learning is mostly used in computer vision and natural language processing tasks like sentiment analysis due to the huge amount of computational power required.The working of transfer learning is shown below 5.1

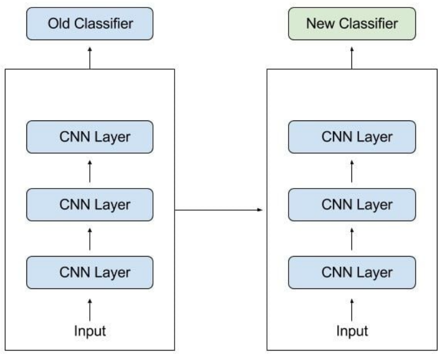


Figure 5.1: Transfer learning [7]

### Why Use Transfer Learning

While there are many advantages to transfer learning, the three primary ones are reduced training time, improved neural network performance (for the most part), and reduced data requirements.

To train a neural network from beginning, a large amount of data is typically required, but access to that data isn't always possible. This is when transfer learning comes in useful. Because the model has previously been pre-trained, transfer learning allows for the construction of strong machine learning models with relatively minimal training data. This is particularly useful for natural language processing, as producing huge labeled data sets mostly requires expert knowledge. Additionally, because it can occasionally take days or even weeks to train a deep neural network from scratch on a demanding job, training time is shortened.

## 5.2 VGG-16

Convolutional neural networks, or CNNs, like the VGG16 model are thought to be among the best computer vision models available today. The model's developers assessed the networks and used an architecture with minuscule (3 × 3) convolution filters to increase the depth, demonstrating a notable advance above previous state-of-the-art setups. They increased the depth to roughly 138 trainable parameters by pushing it to 16–19 weight layers. The architecture is shown in the figure 5.2 below. It showcases in depth the number of layers and the architecture used.

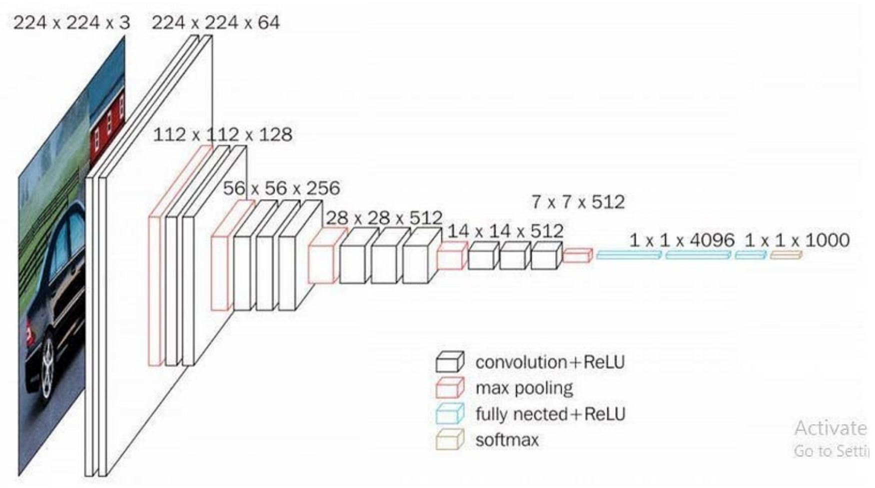


Figure 5.2: VGG architecture [8]

- The sixteen in VGG16 stands for sixteen weighted layers. Although VGG16 contains twenty-one layers total—sixteen convolutional layers, five Max Pooling layers, and three Dense layers—it only has sixteen weight layers, or learnable parameters layers.
- VGG16 takes input tensor size as 224, 244 with 3 RGB channel
- The most distinctive feature of VGG16 is that its convolution layers of a 3x3 filter with stride 1 are its main focus, rather than having a lot of hyper-parameters. The padding and maxpool layer of a 2x2 filter with stride 2 are also always employed.

- After a stack of convolutional layers, three Fully-Connected (FC) layers are placed: the first two have 4096 channels apiece, while the third conducts 1000-way ILSVRC classification, resulting in 1000 channels (one for each class). Soft-max layer is the last layer.

## 5.3 VGG 16 plus Custom Detector

VGG-16 can act as an extremely efficient feature extractor, now after extracting the features, I need to implement a custom detector. But before all of that, I had to decide which colorspace to use, after careful consideration and going through several research papers I concluded that using the CIELAB colorspace would be the most advantageous.

### 5.3.1 CIELAB

Richard Hunter first presented the Lab tri-stimulus model in the 1940s. It is scaled to produce almost uniform spacing between perceived color changes. Hunter's Lab was never legally recognized as an international standard, despite becoming the de facto model for charting absolute color coordinates and color discrepancies. CIELAB is also referred to as L*a*b colorspace, it expresses color as three values

- L is for perceptual lightness
- a* stands for the Red/Green value
- b* stands for the Blue/Yellow Value

It is very useful in detecting small differences in color. This property makes it useful in the realm of image colorization.LAB employs the three channels that stand for ”L” (lightness), ”a” (red/green), and ”b” (blue/yellow), as opposed to RGB, which starts with blackness and adds different degrees of red, green, and blue brightnesses to generate your final pixel color and brightness value. Below figure 5.5 explains how the CIELAB colorspace is depicted and what is the significance of each channel in the colorspace.

#### Intuition behind CIELAB colorspace

We should provide a grayscale image to a model to train it for colorization, and then we should aim for it to produce a colorful result. By giving the model (the grayscale image) the L channel and asking it to forecast the other two channels (*a, *b), we may use L*a*b. Once the model makes its prediction, we concatenate all the

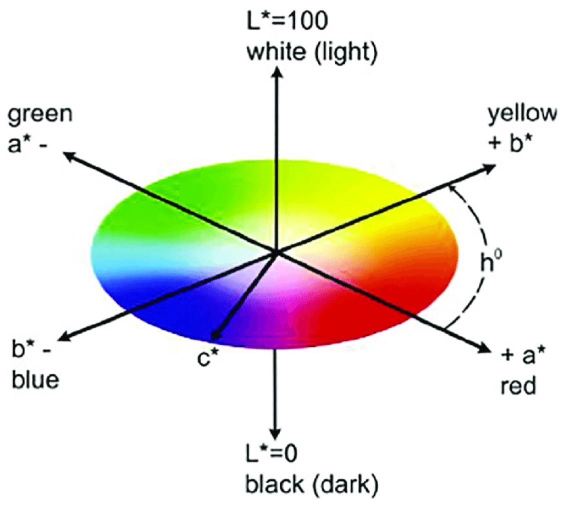


Figure 5.3: LAB colorspace [9]

channels to get our colorful image. However, if you use RGB, you have to first convert the image to grayscale, then feed the model the grayscale image and hope that it will correctly forecast three numbers. This is a much more challenging and unstable operation because there are a lot more possible combinations of three numbers than there are of two.Predicting the three numbers for each pixel involves selecting from $256^3$ combinations, or more than 16 million options if we assume that there are 256 choices (the actual number of choices in an 8-bit unsigned integer image). However, when predicting two numbers, there are roughly 65000 choices.

### 5.3.2 Custom Detector

Our input to the VGG-16 model will be the ”L” channel,but the problem we faced was that VGG-16 has an input shape of (224,224,3) where 224 was the width and the height of the image, and 3 is the number of channels we need to give. In our input we only have 1 input channel which is the ”L” channel, to correct this, we repeat the lightness channel three times.

After predicting the lightness channel from the VGG-16, our custom detector will give us the a and b channels, we will then convert the LAB colorspace to RGB colorspace and then display the image.

In our custom detector, I have used upsamling layers instead of pooling layers to increase the dimensions of the layer.The below figure 6.1 helps us understand how upsampling layers actually work. We an observe that using upsampling layers will help us increase the spatial dimension of the model.

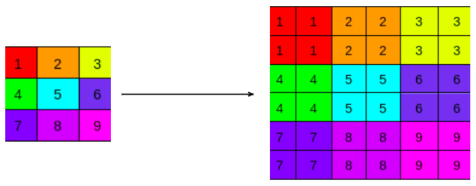


Figure 5.4: Upsampling example [10]

The overall diagram of my model is shown in the figure below 5.5

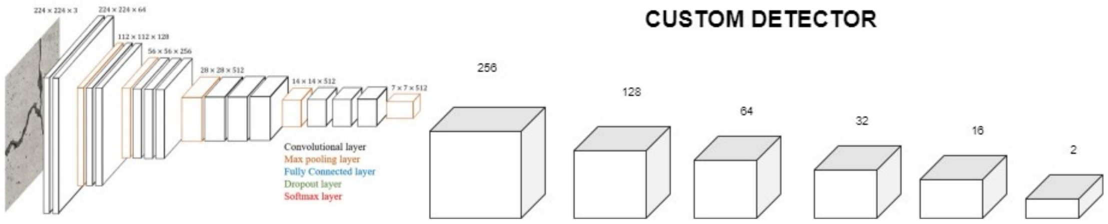


Figure 5.5: Model Architecture [9]

# Chapter 6
# Generative Adversarial Networks

In the realm of image colorization, a more efficient and accurate way is presented using Generative Adversarial Networks.

## 6.1 What are Generative Adversarial Networks

GANs, or Generative Adversarial Networks, are a kind of generative modeling that make use of convolutional neural networks and other deep learning techniques. In generative modeling, regularities or patterns in the input data are automatically found and learned so that the model can be used to generate or output new examples that could have been realistically taken from the original dataset. This is an unsupervised learning task in machine learning. By rephrasing the task as a supervised learning problem with two sub-models—the discriminator model, which attempts to categorize examples as real (from the domain) or fake (generated), and the generator model, which we train to produce new examples—generative models can be cleverly trained using GANs. Together, the two models train in an adversarial zero-sum game until the discriminator model is tricked around half the time, indicating that the generator model is producing examples that make sense.

### 6.1.1 Difference between Discriminative vs Generative Modeling

Discriminative usually comes under supervised learning, as in supervised learning we are interested in developing a model to predict a class label given an example of input variables. This type of predictive modeling task is called classification which is traditionally known as discriminative modeling. The figure 6.1 below shows how discriminative models are configured.

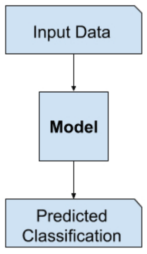


Figure 6.1: Discriminative Modeling [11]

Unsupervised models summarize the distribution of input variables that may be used to create or generate new examples in the input distribution.Such models are known as generative models as shown in the figure 6.2 below.

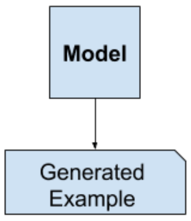


Figure 6.2: Generative Model [11]

## 6.2 GAN architecture

Two sub-models make up the architecture of the GAN model: a generator model that creates new instances, and a discriminator model that determines whether the examples generated by the generator model are authentic, coming from the domain, or fraudulent.

**Generator Model**

The generator model creates a sample in the domain by using a fixed-length random vector as input. The generative process is seeded with a vector that is randomly selected from a Gaussian distribution. A compressed representation of the data distribution will be formed once training is complete, with points in this multidimensional vector space corresponding to points in the issue domain.
Latent variables make up this vector space, which is also known as a latent space. The factors that are significant for a domain but are not readily visible are known as latent or hidden variables.
Latent variables, also known as a latent space, are frequently described as a projection or compression of a data distribution. In other words, a latent space offers a high-level idea or compression of the observed raw data, such as the distribution of the input data. With GANs, fresh points extracted from the latent space can be fed into the generator model as input and utilized to produce novel and distinct output examples. This is because the generator model gives meaning to points in a selected latent space.

### Discriminator Model

The discriminator model predicts a binary class label of real or fake (created) based on an input sample from the domain that can be either real or generated. The training dataset is where the actual example is found. The generator model produces the generated examples. The discriminator is a typical classification paradigm that is widely known. We are more interested in the generator, therefore we reject the discriminator model after the training phase. As the generator has gotten better at efficiently extracting characteristics from examples in the issue domain, it can occasionally be used for new purposes. Using the same or comparable input data, transfer learning applications can take advantage of some, all, or even all of the feature extraction layers. The figure 6.3 shows how a complete architecture of Generative Adversarial Network looks.

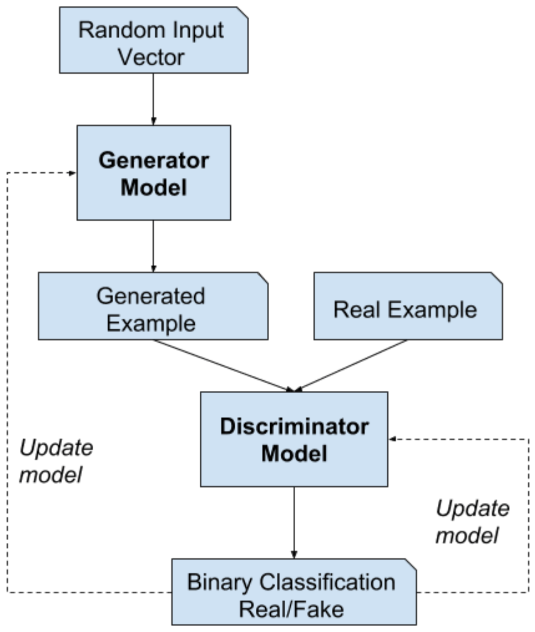


Figure 6.3: Complete GAN architecture [11]

## 6.2.1 Conditional GANs

The application of GANs for conditionally generating an output is a significant extension. When the input, the random vector from the latent space, is given (conditioned by) some more input, the generative model can be trained to produce new examples from the input domain. A class value, such male or female when creating pictures of individuals, or a digit when creating pictures of handwritten numerals, could be the supplementary input. Additionally, the discriminator is conditioned, which means that it receives the additional input in addition to an input image that can be real or phony. The discriminator would therefore anticipate that the input would be of that class in the case of a classification label type conditional input, teaching the generator to produce samples of that class in order to trick the discriminator. This is one method of generating examples from a domain of a given kind using a conditional GAN. To go one step further, the GAN models can be trained on a domain example, such an image. This makes it possible to use GANs for tasks like image-to-image or text-to-image translation.

## 6.3 Image Colorization Using Conditional GANs

Colorization was one of the many image-to-image tasks in deep learning for which a generic solution was presented in the Image-to-Image Translation using Conditional Adversarial Networks study, also known as pix2pix [20]. This method uses two types of losses: an adversarial (GAN) loss that aids in solving the problem in an unsupervised way and an L1 loss that turns the task into a regression.

In our setting, the generator model takes a grayscale image which is a 1-channel image, and produces a 2-channel image ( a channel for a* and another for b*). The discriminator, takes these two produced channels and concatenates them with the input grayscale image and decides whether the 3-channel image is fake or real. Of course the discriminator also needs to see some real images(in LAB colorspace) that are not produced by the generator and should learn whether they are real or not.

The condition that we impose is the grayscale image that both the generator and discriminator see. We expect our model to take this condition into consideration.

Consider x as the grayscale image, z as the input noise for the generator and y as the 2 channel output we want from the generator. G is the generator model and D is the discriminator.
Loss for our conditional GAN will be:-
**L(*G*,*D*)=** $E_{x,y}[logD(x,y)] + E_{x,z}[log(1 - D(x, G(x,z))]$ this loss function helps to produce good-looking colorful images that seem real, to further help the models, we combine the loss function with L1 loss of the predicted colors compared with the actual colors:
$L_{L1}(G) = E_{x,y,z}[||y - G(x, z)||_1]$ In the event that we only use L1 loss, the model will still learn to colorize the images, but it will do so conservatively and typically use colors like ”gray” or ”brown” because, in situations where it is unsure of the optimal color, it takes the average and uses these colors to minimize L1 loss (this is comparable to the blurring effect of L1 or L2 loss in super-resolution task). Also, because it lessens the effect of creating grayscale images, the L1 loss is chosen over the L2 loss (or mean squared error).
The combined loss function will be:-
$\mathbf{G}^* = argmin(G)max(D)L_{cGAN}(G, D) + \lambda_{L1}(G)$

lambda is the coefficient to balance the contributions of the two losses to the final loss.

In the paper referred to, for the generator model they used a U-Net architecture. It's crucial to realize that, in creating the U-Net, the middle module is the starting point. At each iteration, down- and up-sampling modules are added to the left and right of the middle module, respectively, until the input and output modules are reached.The generator model architecture is shown in the figure 6.4 below. The sequence in

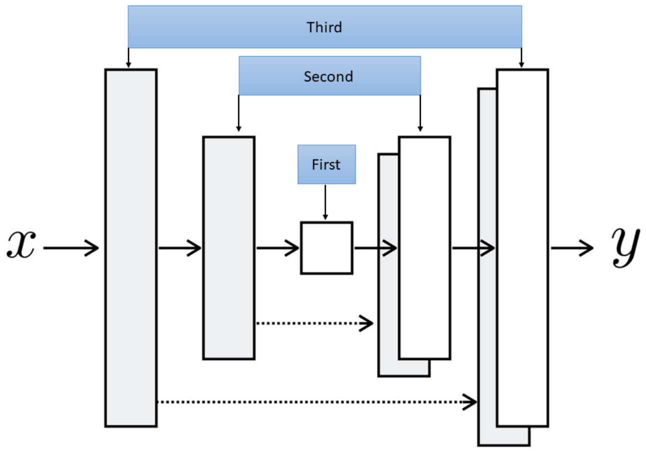


Figure 6.4: Generator Model [11]

which the associated modules are constructed using the code is indicated by the blue rectangles. Although this image only shows a portion of the U-Net, it gives you an idea of what it looks like. It's also important to note that the algorithm descends eight layers. This means that if we begin with a 256 by 256 image, we will receive a 1 by 1 (256 / $2^8$) image in the middle of the U-Net, which will then be up-sampled to create a 256 by 256 image (with two channels).

For the Discriminator model, Conv-BatchNorm-LeackyReLU blocks are stacked to create a model that determines if the input image is real or fake. A patch Discriminator is used, For each patch of the input image, say 70 by 70 pixels, the model generates a number, and for each one, it determines whether or not it is a fake. It makes sense to use a model like this for the colorization work because the model must make significant local adjustments and may need to decide on the entire image because a vanilla discriminator may not be able to handle the nuances of this assignment. We run the module's forward method first and only once every iteration (batch of the training set), storing the results in the class's fake color property.

Next, we use the backward D approach to train the discriminator by feeding it the fake images that the generator produced, labeling them as fake, and making sure they are detached from the generator's graph so they function as a constant like normal images. Next, we give the discriminator a batch of genuine photos from the training set and mark them as real. The two losses—fake and real—are added up, the average is calculated, and the total loss is called backward. The generator can now be trained. The backward G approach involves feeding a fictitious image to the discriminator, trying to trick it by giving it actual labels and figuring out the adversarial loss. As previously stated, we also employ L1 loss. To balance the two losses, we calculate the separation between the target and anticipated two channels, multiply the loss by a coefficient (in our example, 100), and add the result to the adversarial loss. Next, we initiate the loss's backward method call.

# Chapter 7
# Results

The model was trained using the VGG-16 model and using a custom detector made of a convolutional neural network.

## 7.1 Dataset

There were many available datasets but due to limitations present in computational power, I decided to create my own datasets. To explore the variations, I created three datasets. All the images were taken from Google Chrome using an extension called *'Image Downloader'*.

- 1. Dataset1 containing nature, landscapes which held 728 images
- 2. Dataset2 containing pictures of humans which held 545 images
- 3. Dataset3 concatenates Dataset1 and Dataset2 and it contains 1273 images

All of these datasets were trained using the same model

Table 7.1: Model Specification

| | |
|---|---|
| Optimizer | Adam |
| Loss Function | Mean Squared Error |
| Metrics | Accuracy |
| Number of Epochs | 2000 |
| Batch Size | 64 |

The results corresponding to the training were:-

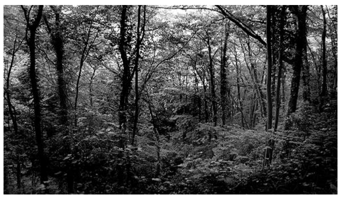
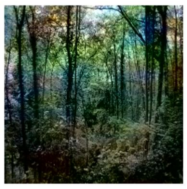
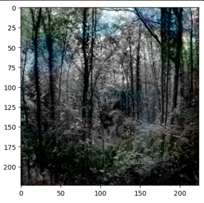
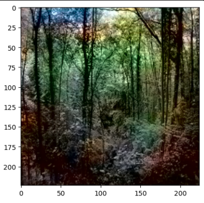
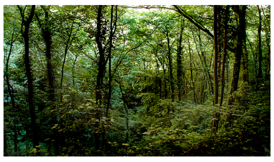
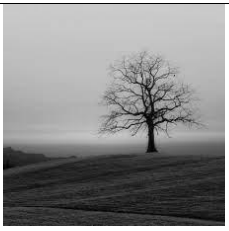
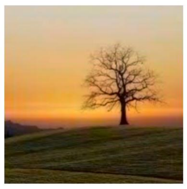
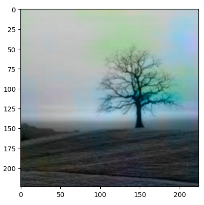
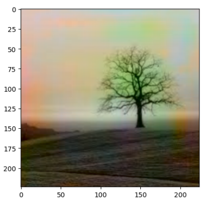
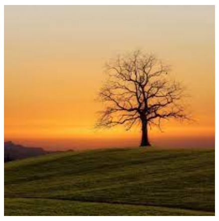
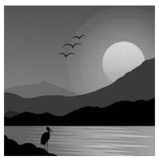
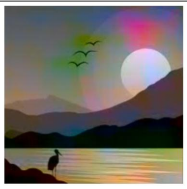
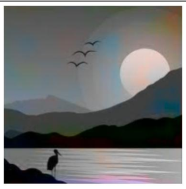
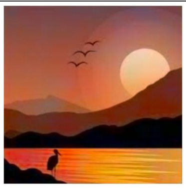
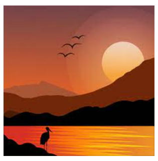
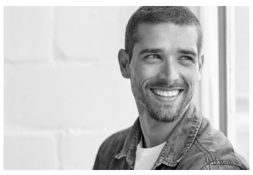
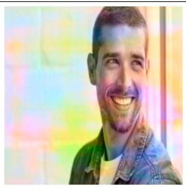
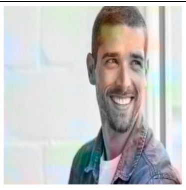
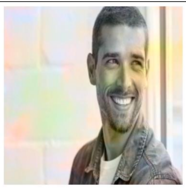
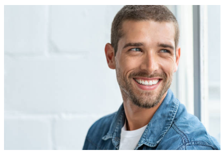
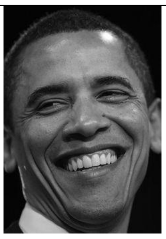
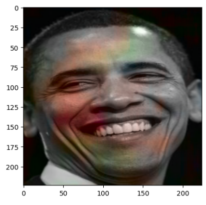
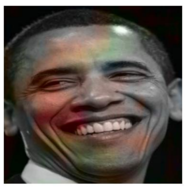
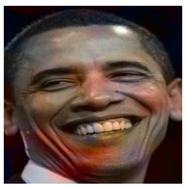
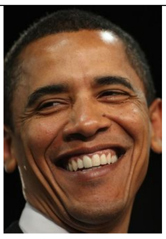
Table 7.2: Dataset vs Accuracy

| **Datasets** | **Accuracy** |
|---|---|
| Dataset 1 | 91.2% |
| Dataset 2 | 73% |
| Dataset 3 | 89.19% |

## 7.2 Predicted Outputs

Below are examples of grayscale image, predicted image and the ground truth

Table 7.3: Results

| GrayScale Input | Dataset1 | Dataset2 | Dataset3 | Ground Truth |
|---|---|---|---|---|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Inference**

We can notice that overall the model containing dataset3 gives us more accurate predictions. This is definitely due to the fact that the dataset is larger and hence it has more data to learn from. Model trained on dataset2 hardly gives the correct output when a nonhuman image is given while dataset1 gives better performance on human and non-human images.

## 7.3 Summary

Image colorization is a compelling area of research and application within computer vision. bridges the gap between historical grayscale imagery and the vibrant, colorful representations we encounter in our everyday visual experiences, with significant implications for fields such as art restoration, film colorization, and enhancing the visual appeal of old or historical photographs.

Given the complexity of the problem at hand, I delved into the realm of Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs). This helped me understand how these tools work and how I could use them to my advantage. I also learned about transfer learning, which is gaining popularity in a world where not everyone has access to expensive computing resources. This approach allows for improved performance in tasks, even with limited resources, by leveraging pre-trained models. It's a significant development in the field.

## 7.4 Future Scope

In pursuit of my research objectives, I plan to implement the current dataset leveraging the capabilities of a Conditional Generative Adversarial Network (GAN), as shown in chapter 6. This methodology holds the promise of yielding better performance metrics, thereby enhancing the overall effectiveness of the model.

Furthermore, an alternative way involves the utilization of the comprehensive ImageNet Dataset. This dataset will furnish the model with a more robust and comprehensive training regimen, leading to better performance.

Envisioning the integration of Conditional GAN in conjunction with the ImageNet dataset, I aspire to attain an accuracy threshold surpassing 98%. This targeted benchmark is integral to realizing the ultimate objective of deploying the model on a website or mobile application.

# References

[1] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, 2018.

[2] (Accessed 2023) tanh activation function. [Online]. Available: https://paperswithcode.com/method/tanh-activation

[3] (Accessed 2023) Cnn architecture. [Online]. Available: https://github.com/Jay22K/devanagari-character-recognition

[4] (Accessed 2023) Cnn. [Online]. Available: https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[5] (Accessed 2023) autoencoder architecture. [Online]. Available: https://www.compthree.com/blog/autoencoder/

[6] (Accessed 2023) autoencoder. [Online]. Available: https://iq.opengenus.org/types-of-autoencoder

[7] (Accessed 2023) transfer learning. [Online]. Available: https://builtin.com/data-science/transfer-learning

[8] (Accessed 2023) Vgg-16. [Online]. Available: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

[9] M. Sewak, S. Sahay, and H. Rathore, "An overview of deep learning architecture of deep neural networks and autoencoders," *Journal of Computational and Theoretical Nanoscience*, vol. 17, pp. 182–188, 01 2020.

[10] (Accessed 2023) Upsampling. [Online]. Available: https://medium.com/@suryatejamenta/upsampler-and-convolution-transpose-layers-7e4346c05bb6

[11] (Accessed 2023) Gan. [Online]. Available: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

[12] S. Anwar, M. Tahir, C. Li, A. Mian, F. S. Khan, and A. W. Muzaffar, "Image colorization: A survey and dataset," *arXiv preprint arXiv:2008.10774*, 2020.

[13] J. Hwang and Y. Zhou, "Image colorization with deep convolutional neural networks," in *Stanford University, Tech. Rep.*, 2016.

[14] V. Singh, D. Arora, and P. Sharma, “Image colorization using deep convolution autoencoder,” in *Proceedings of International Conference on Recent Trends in Computing: ICRTC 2021*. Springer, 2022, pp. 431–441.

[15] K. Nazeri, E. Ng, and M. Ebrahimi, “Image colorization using generative adversarial networks,” in *Articulated Motion and Deformable Objects: 10th International Conference, AMDO 2018, Palma de Mallorca, Spain, July 12-13, 2018, Proceedings 10*. Springer, 2018, pp. 85–94.

[16] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[17] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.

[18] T. Welsh, M. Ashikhmin, and K. Mueller, “Transferring color to greyscale images,” in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 277–280.

[19] A. Levin, D. Lischinski, and Y. Weiss, “Colorization using optimization,” in *ACM SIGGRAPH 2004 Papers*, 2004, pp. 689–694.

[20] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer, 2016, pp. 649–666.