

# **IMAGE COLORIZATION AND ENHANCEMENT USING GENERATIVE ADVERSARIAL NETWORKS**

**Capstone Project Report**

**Panel Evaluation**

**Submitted by:**

**NIMISH MATHUR (101803323)  
JADI AKHILESH PRASAD (101803188)  
SHUBHAM GOENKA (101803223)  
DEV KEVLANI (101803294)**

**BE Fourth Year, Computer Engineering CPG No: 62**

Under the Mentorship of

Dr. Surjit Singh

Assistant Professor

Dr. Singara Singh

Associate Professor



**Computer Science and Engineering Department**

**TIET, Patiala**

**December 2021**

## ABSTRACT

---


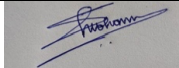
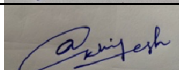
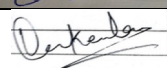
Automatic synthesis of optimized/converted images from an input image would be interesting and useful, but current AI systems are still far from this goal. However, in recent years generic and powerful recurrent neural network architectures have been developed to learn discriminative image feature representations. Meanwhile, deep convolutional generative adversarial networks (GANs) have begun to generate highly compelling images of specific categories, such as faces, album covers, and room interiors. GAN is a framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model  $G$  that captures the data distribution, and a discriminative model  $D$  that estimates the probability. This framework corresponds to a minimax two-player game. In this work, we develop a novel deep architecture and GAN formulation to effectively bridge these advances in input image and image modeling. We demonstrate the capability of our model to generate optimized images from train image.

## DECLARATION

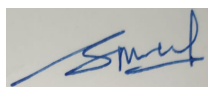
---

We hereby declare that the design principles and working prototype model of the project entitled Image Colorization and Enhancement using GANs is an authentic record of our own work carried out in the Computer Science and Engineering Department, TIET, Patiala, under the guidance of Dr. Surjit Singh and Dr. Singara Singh during 6<sup>th</sup> & 7<sup>th</sup> semester (2021).

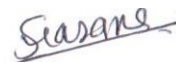
Date: 17-12-2021

Roll No.	Name	Signature
101803323	Nimish Mathur	
101803223	Shubham Goenka	
101803188	Jadi Akhilesh Prasad	
101803294	Dev Kevlani	

Counter Signed By:



Mentor:  
Dr. Surjit Singh  
Designation: Assistant Professor  
Computer Science & Engineering Department  
TIET, Patiala



Co-Mentor:  
Dr. Singara Singh  
Designation: Associate Professor  
Computer Science & Engineering Department  
TIET, Patiala

## ACKNOWLEDGEMENT

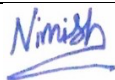
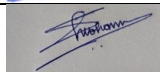
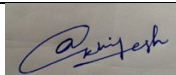
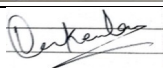
---

We would like to express our thanks to our mentors Dr. Surjit Singh and Dr. Singara Singh. They have been of great help in our venture, and an indispensable resource of technical knowledge. They are truly amazing mentors to have.

We are also thankful to Dr. Maninder Singh, Head, Computer Science and Engineering Department, entire faculty and staff of Computer Science and Engineering Department, and also our friends who devoted their valuable time and helped us in all possible ways towards successful completion of this project. We thank all those who have contributed either directly or indirectly towards this project.

Lastly, we would also like to thank our families for their unyielding love and encouragement. They always wanted the best for us and we admire their determination and sacrifice.

Date: 17-12-2019

Roll No.	Name	Signature
101803323	Nimish Mathur	
101803223	Shubham Goenka	
101803188	Jadi Akhilesh Prasad	
101803294	Dev Kevlani	

# TABLE OF CONTENTS

---

<b>ABSTRACT</b>	<b>ii</b>
<b>DECLARATION</b>	<b>iii</b>
<b>ACKNOWLEDGEMENT</b>	<b>iv</b>
<b>LIST OF TABLES</b>	<b>viii</b>
<b>LIST OF FIGURES</b>	<b>ix</b>
<b>LIST OF ABBREVIATIONS</b>	<b>x</b>
<b>CHAPTER 1- INTRODUCTION</b>	<b>1</b>
1.1 PROJECT OVERVIEW	1
1.1.1 TECHNICAL TERMINOLOGY	1
1.1.2 PROBLEM STATEMENT	2
1.1.3 GOAL	3
1.1.4 SOLUTION	3
1.2 NEED ANALYSIS	3
1.3 RESEARCH GAPS	4
1.4 PROBLEM DEFINITION AND SCOPE	5
1.5 ASSUMPTIONS AND CONSTRAINTS	5
1.6 APPROVED OBJECTIVES	6
1.7 METHODOLOGY USED	7
1.8 PROJECT OUTCOMES AND DELIVERABLES	8
1.9 NOVELTY OF WORK	8
<b>CHAPTER 2 - REQUIREMENT ANALYSIS</b>	<b>9</b>
2.1 LITERATURE SURVEY	9
2.1.1 THEORY ASSOCIATED WITH PROBLEM AREA	9
2.1.2 EXISTING SYSTEMS AND SOLUTIONS	9
2.1.3 RESEARCH FINDINGS FOR EXISTING LITERATURE	10
2.1.4 THE PROBLEM THAT HAS BEEN IDENTIFIED	13
2.1.5 SURVEY OF TOOLS AND TECHNOLOGIES USED	13
2.2 STANDARDS	14
2.3 SOFTWARE REQUIREMENTS SPECIFICATION	15
2.3.1 INTRODUCTION	15

2.3.1.1 PURPOSE	15
2.3.1.2 INTENDED AUDIENCE	15
2.3.1.3 PROJECT SCOPE	16
2.3.2 OVERALL DESCRIPTION	16
2.3.2.1 PRODUCT PERSPECTIVE	16
2.3.2.2 PRODUCT FEATURES	16
2.3.3 EXTERNAL INTERFACE REQUIREMENTS	17
2.3.3.1 USER INTERFACES	17
2.3.3.2 HARDWARE INTERFACES	18
2.3.3.3 SOFTWARE INTERFACES	18
2.3.4 REQUIREMENTS	18
2.3.4.1 FUNCTIONAL REQUIREMENTS	19
2.3.4.2 NON-FUNCTIONAL REQUIREMENTS	19
2.3.4.3 PERFORMANCE REQUIREMENTS	20
2.3.4.3.1 SIMPLICITY	20
2.3.4.3.2 AVAILABILITY	20
2.3.4.3.3 MAINTAINABILITY	20
2.4 RISK ANALYSIS	21
<b>CHAPTER 3 – METHODOLOGY ADOPTED</b>	<b>22</b>
3.1 INVESTIGATIVE TECHNIQUES	22
3.2 PROPOSED SOLUTION	22
3.3 WORK BREAKDOWN STRUCTURE	23
3.4 TOOLS AND TECHNOLOGIES USED	24
<b>CHAPTER 4 - DESIGN SPECIFICATIONS</b>	<b>25</b>
4.1 SYSTEM ARCHITECTURE	25
4.2 DESIGN LEVEL DIAGRAMS	27
4.3 USER INTERFACE DIAGRAMS	31
<b>CHAPTER 5 – IMPLEMENTATION AND EXPERIMENTAL RESULTS</b>	<b>36</b>
5.1 EXPERIMENTAL SETUP (OR SIMULATION)	36
5.2 EXPERIMENTAL ANALYSIS	36
5.2.1 DATA	36

5.2.2 PERFORMANCE PARAMETERS	37
5.3 WORKING OF PROJECT	37
5.4 TESTING PROCESS	44
5.4.1 TEST PLAN	44
5.4.1.1 FEATURES TO BE TESTED	44
5.4.1.2 TEST STRATEGY	45
5.4.1.3 TEST TECHNIQUES	45
5.4.2 TEST CASES	45
5.4.3 TEST RESULTS	45
5.5 RESULTS AND DISCUSSIONS	46
5.6 INFERENCES DRAWN	46
5.7 VALIDATION OF OBJECTIVES	46
<b>CHAPTER 6: CONCLUSION AND FUTURE WORK DIRECTIONS</b>	<b>48</b>
6.1 CONCLUSIONS	48
6.2 ENVIRONMENTAL, ECONOMIC AND SOCIETAL BENEFITS	48
6.3 REFLECTIONS	48
6.4 FUTURE WORK	49
<b>CHAPTER 7: PROJECT METRICS</b>	<b>50</b>
7.1 CHALLENGES FACED	50
7.2 RELEVANT SUBJECTS	51
7.3 INTERDISCIPLINARY KNOWLEDGE SHARING	52
7.4 PEER ASSESSMENT MATRIX	53
7.5 ROLE PLAYING AND WORK SCHEDULE	53
7.6 STUDENT OUTCOMES DESCRIPTION AND PERFORMANCE INDICATORS (A-K MAPPING)	55
7.7 BRIEF ANALYTICAL ASSESSMENT	59
<b>APPENDIX A: REFERENCES</b>	<b>61</b>
<b>APPENDIX B: PLAGIARISM REPORT</b>	<b>63</b>

## LIST OF TABLES

---

Table No.	Caption	Page No.
Table 1	Research Findings and existing literature	11
Table 2	Software Engineering Standards	15
Table 3	Investigative Techniques	22
Table 4	Objective validation table	46
Table 5	Relevant subjects	51
Table 6	Peer assessment matrix	53
Table 7	Individual roles of group members	53
Table 8	Work Schedule	54
Table 9	A-K mapping	55



## LIST OF FIGURES

---

Figure No.	Caption	Page No.
Figure 1	Generator and discriminator networks	2
Figure 2	Work Breakdown Structure	23
Figure 3	MVC architecture	25
Figure 4	3 tier architecture	26
Figure 5	User Interface Diagram	26
Figure 6	Data Model Diagram	27
Figure 7	Flowchart	28
Figure 8	Sequence Diagram	29
Figure 9	E-R Diagram	30
Figure 10	Use Case Diagram	31
Figure 11	CIFAR-10 Dataset	36
Figure 12	UNET Architecture (256*256)	38
Figure 13	System Component Diagram	41
Figure 14	Training DCGAN	42
Figure 15	Fake Images generated	42
Figure 16	Real Images	43
Figure 17	Fake Images (Batch-8)	43
Figure 18	Real Images (Batch-8)	43
Figure 19	Image Colorization of Uploaded Image	44

## LIST OF ABBREVIATIONS

---

<b>CNN</b>	Convolutional Neural Network
<b>GAN</b>	Generative Adversarial Neural Network
<b>RNN</b>	Recurrent Neural Network

## 1.1 Project Overview

### 1.1.1 Technical Terminology

The problem of generating images from input raw images has recently gained interest in the research community and it has various potential applications, such as photo editing, video generation and digital design and the field is quite wide for its application. In recent researches, the expressiveness of deep convolutional and recurrent networks enables image generation tasks significantly comparing with the traditional attribute representation approaches. Moreover, the recently developed Conditional Generative Adversarial Networks [1] (ConditionalGAN) model has demonstrated its capability in generating high-resolution images with photo-realistic details. Basically, Generative adversarial networks (GANs) are a class of artificial intelligence algorithms used in unsupervised machine learning, implemented by a system of two neural networks contesting with each other in a zero-sum game framework [2]. One network generates candidates and the other evaluates them. Typically, the generative network learns to map from a latent space to a particular data distribution of interest, while the discriminative network discriminates between instances from the true data distribution and candidates produced by the generator. Such framework is shown in Figure 1. The generative network's training objective is to increase the error rate of the discriminative network.

However, it is very difficult to train GAN to generate high-resolution photo-optimized images from input image descriptions. Simply adding more up sampling layers in state-of-the-art GAN models for generating high-resolution images generally results in training instability and produces nonsensical outputs. The main difficulty for generating high-resolution images by GANs is that supports of natural image distribution and implied model distribution may not overlap in high dimensional pixel space. This problem is more severe as the image resolution increases.

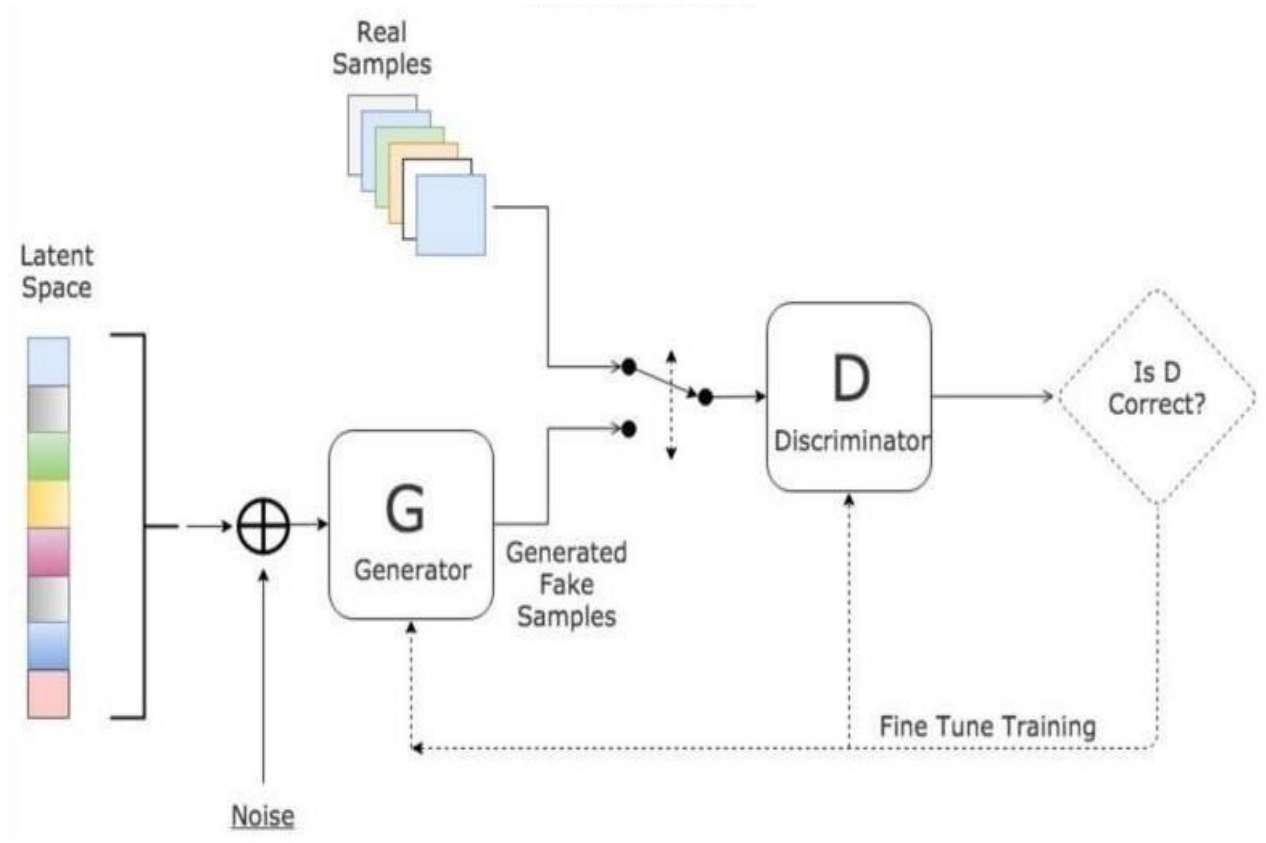


Figure 1: Generator and discriminator networks

### 1.1.2 Problem Statement

Over the past decade, the process of automated image colorization has been of significant interest to many application areas including restoration of aged or degraded images. This problem is highly incurable due to the large degree of freedom during the assignment of color information. Several recent developments in automated colorization include images that have a common theme or require highly processed data such as semantic maps as input. In our approach, we attempt to generalize the colorization process as a whole by using a Deep Convolutional Generative Adversarial Network (DCGAN).

### **1.1.3 Goal**

Image colorization is an image-to-image translation problem that maps a high dimensional input to a high dimensional output. It can be seen as pixel-wise regression problem [6] where structure in the input is highly aligned with structure in the output. That means the goal is to construct a network that need not only generate an output with the same spatial dimension as the input, but also provide color information to each pixel in the grayscale input image.

### **1.1.4 Solution**

The solution to the problem of automated image colorization and restoration of aged or degraded images is construction of a model of conditional GAN. In a traditional GAN, the input of the generator is randomly generated noise data. However, this approach is not applicable to the automatic colorization problem because grayscale images serve as the inputs of our problem rather than noise. This problem was addressed by using a variant of GAN called conditional generative adversarial networks. Since no noise is introduced, the input of the generator is treated as zero noise with the grayscale input as a prior.

## **1.2 Need Analysis**

Gray scale to colored image is usually visualized as a person interesting or clear image which is not look good before, either manually created or clicked by a human. If generation of new images is possible, computer vision tasks will never fall short of sufficient data. Even when specific data will be required, more data could be generated which is equivalently good for discriminative analysis using the data that is already present.

A few more areas of application could be: -

1. A brain imagines things in the form of images. Generation of images through GANs can prove to be that part of the brain for the future AI.
2. Restoration of degraded images.
3. Restoration of vintage images and black and white images/film reel.
4. Night Vision camera images/videos to be colored and improved using this model.

5. Formation of media without the need of implementing everything through explicit tools can decrease huge workload.

- Photo Editing: Filtering and editing of images is a very tedious task and can be easily done using related images.
- Object Coloring: Imagine scientists specifying every bit as image to the model, even fine coloring

### **1.3 Research Gaps**

Research gap refers to a knowledge gap that is yet to be researched. As it was a computer science project, the research gaps were vast and integrating the various knowledge that we had acquired in the various fields was even tougher. A large number of subjects that we have studied during our bachelors were being applied to project and hence revisiting their basics was of the most basic importance.

Moving on to the base station, since conversion of images had been used to generate a possible image. Computer Vision had to be given emphasis. Computer vision techniques included normalization etc. Deep learning concepts such as CNN and ANN were used to generate the output.

The project required a better understanding and good implementation of the following but was not limited to:

1. Image Processing (Normalization)
2. Deep Learning:
  - a. Convolutional Neural Networks
  - b. Artificial Neural Networks
  - c. Batch Normalization
  - d. Dropout
  - e. Max Pooling
  - f. Adam Optimization/Gradient Descent

## **1.4 Problem Definition and Scope of Project**

### **Problem Definition**

In recent years generic and powerful recurrent neural network architectures have been developed to learn discriminative text feature representations. Over the past decade, the process of automated image colorization has been of significant interest to many application areas including restoration of aged or degraded images. This problem is highly incurable due to the large degree of freedom during the assignment of color information. Several recent developments in automated colorization include images that have a common theme or require highly processed data such as semantic maps as input. In our approach, we attempt to generalize the colorization process as a whole by using a DCGAN.

### **Scope**

1. At the current research point, the images are visually plausible only if the output is taken as a lower resolution. Higher resolution could create a whole new dimension in the use case.
2. Given limited GPU memory and resources, the training can only be done on a smaller dataset focusing on a few topics at a time. The complexity of the situation could be handled once the resources are available in the coming years. The project has the ability to act as an imagination unit for an AI-powered humanoid.

## **1.5 Assumptions & Constraints**

### **Assumptions**

- User should know about the dataset on which our model is trained on and gives the input image accordingly.
- We assume that we will get the permission to use these computers for training our models.

- The customer will have basic knowledge of computer science and image processing.
- Behavior of deep learning models outperforms any other models or methodology.
- Selected model is the best one for generation of images.
- GPU computations using CUDNN are optimized & there is no need for further manual optimizations.
- Keras/PyTorch would be enough to implement the model and there will be no need for incisive tweaking using other deep learning frameworks.

### **Constraints**

- Basic models CNNs can't be noise proof, with images having noise. With random noise addition output confidence may change. Hence, an assumption of handling such noise is avoided instead focus on simple cases is done.
- GANs are highly unstable while working with high resolution images.
- Version consistency amongst all modules of code, to implement uniformity for each one of them in terms of coding standards.

## **1.6 Approved Objectives**

- The goal is to construct a network that need not only generate an output with the same spatial dimension as the input, but also provide color information to each pixel in the grayscale input image.
- Training a deep convolutional GAN to generate realistic images from greyscale images. The result will show that generated images are consistent with input images.



## 1.7 Methodology

### 1.7.1 High level Architecture

**Deep Convolutional Neural Network** [4]: CNNs use a variant of multiple layer perceptron's designed to require negligible pre-processing. They are named shift invariant or space invariant artificial neural networks, as per their shared-weights framework and translation invariance features. The idea of CNNs is used in integration with the concept of GANs.

**Generative Adversarial Networks:** A discriminative model studies a function that maps the input data ( $x$ ) to a wanted resultant class label ( $y$ ). In relative terms, they straight away study the conditional distribution  $P(y|x)$ . A generative model aims to study the joint probability of the input data and labels concurrently, i.e.  $P(x,y)$ . This is eventually changed to  $P(y|x)$  for classification using Bayes rule but the generative skill might have other uses, like generating new ( $x, y$ ) samples.

### 1.7.2 Low Level Architecture

**Batch Normalization:** Spreading of each layer's inputs varies throughout training, as the parameters of the earlier layers vary. This reduces down the training speed by needing lower learning rates as well as parameter initialization, and makes it especially difficult to train models having saturating nonlinearities. The Batch Normalization phenomenon acts as inner covariate shift, and looks at the problem by normalizing layer inputs. The method has its pros by having normalization a part of the model architecture while also executing the normalization for each training mini-batch.

**Dropout:**[8] Dropout is a regularization method used to reduce the overfitting problem in neural networks by avoiding complex co-adaptations on the training data. It is an effective way of executing model averaging with neural networks. The term "dropout" is used as a reference to the dropping out units.

## **1.8 Project Outcomes/Deliverables**

Our project will be able to perform the following tasks:

- Enhanced capability and understanding of adversarial networks as generative models of machine learning.
- Building of dataset to cover all aspects of required domain.
- Automatic synthesis of colored images from greyscale images having human level accuracy is achieved.

## **1.9 Novelty of Work**

- Previous implementations of colorization have poor quality outputs. Our proposed model uses DCGAN's which turned out to give better-quality, high-resolution outputs.
- The output of GANs is highly specific with regard to the dataset used. We used our own dataset and the output was equivalent to previously proposed GAN outputs.
- We also tested our model by training it on the dataset of cars, animals and airplanes achieving sufficient accuracy.

## 2.1 Literature Survey

Luckily, deep learning has empowered enormous progress in the proposed problems – generative adversarial networks and image generation, we keep it as our base and make our project and tackle both the problems.

### 2.1.1 Theory Associated with Problem Area

**GANs** consist of a generator  $G$  and a discriminator  $D$  that compete in a two-player minimax game [7]: The discriminator tries to distinguish real training data from synthetic images, and the generator tries to fool the discriminator. Concretely,  $D$  and  $G$  play the following game as described by the given

$$\min_G \max_D V(D, G)$$

$$V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

below equation.

Goodfellow et al. (2014) prove that this min max game has a global optimum precisely when  $p_z = p_{data}$ , and that under mild conditions (e.g.,  $G$  and  $D$  have enough capacity)  $p_z$  converges to  $p_{data}$ . In practice, in the start of training samples from  $D$  are extremely poor and rejected by  $D$  with high confidence. It has been found to work better in practice for the generator to maximize  $\log(D(G(z)))$  instead of minimizing  $\log(1 - D(G(z)))$ .

**GANs** In a traditional GAN, the input of the generator is randomly generated noise data  $z$ . However, this approach is not applicable to the automatic colorization problem because grayscale images serve as the inputs of our problem rather than noise. This problem was addressed by using a variant of GAN called conditional generative adversarial networks [3]. Since no noise is introduced, the input of the generator is treated as zero noise with the grayscale input as a prior, or

mathematically speaking,  $G(0z|x)$  [9]. In addition, the input of the discriminator was also modified to accommodate for the conditional network. By introducing these modifications, our final cost functions are as follows:

$$\begin{aligned}\min_{\theta_G} J^{(G)}(\theta_D, \theta_G) &= \min_{\theta_G} -\mathbb{E}_z [\log(D(G(\mathbf{0}_z|x)))] + \lambda \|G(\mathbf{0}_z|x) - y\|_1 \\ \max_{\theta_D} J^{(D)}(\theta_D, \theta_G) &= \max_{\theta_D} (\mathbb{E}_y [\log(D(y|x))] + \mathbb{E}_z [\log(1 - D(G(\mathbf{0}_z|x)|x))])\end{aligned}$$

The discriminator gets colored images from both generator and original data along with the grayscale input as the condition and tries to decide which pair contains the true colored image.

## 2.1.2 Existing Systems & Solutions

**Image Colorization Techniques:** Models for the colorization of grayscales began back in the early 2000s. In 2002, Welsh et al. [2] proposed an algorithm that colorized images through texture synthesis. Colorization was done by matching luminance and texture information between an existing color image and the grayscale image to be colorized. However, this proposed algorithm was defined as a forward problem, thus all solutions were deterministic. Levin et al. [4] proposed an alternative formulation to the colorization problem in 2004. This formulation followed an inverse approach, where the cost function was designed by penalizing the difference between each pixel and a weighted average of its neighboring pixels. Both of these proposed methods still required significant user intervention which made the solutions less than ideal.

**Image to Optimized Image Synthesis with Generative Adversarial Networks:** Synthesizing high-quality images from input image is a challenging problem in computer vision and has many practical applications. Samples generated by existing input image to image approaches can roughly reflect the meaning of the given descriptions, but they fail to contain necessary details and vivid object parts. But Generative Adversarial Networks are able to generate 256 256 photo-realistic images conditioned on input image. The GAN takes results and input image as inputs, and

generates images with photo-realistic details. Extensive experiments and comparisons with state-of-the-arts on benchmark datasets demonstrate that the proposed method achieves significant improvements on generating photo-realistic images conditioned on input image.

**ColorUNet: A convolutional classification approach to colorization:** This model tackles the challenge of colorizing grayscale images. It takes a deep convolutional neural network approach, and choose to take the angle of classification, working on a finite set of possible colors. Similarly, to a recent paper, it implements a loss and a prediction function that favor realistic, colorful images rather than “true” ones. It shows that a rather lightweight architecture inspired by the U-Net [10], and trained on a reasonable number of pictures of landscapes, achieves satisfactory results on this specific subset of pictures. It shows that data augmentation significantly improves the performance and robustness of the model, and provide visual analysis of the prediction confidence.

### 2.1.3 Research Findings for Existing Literature

The research done by individual members is described in table 1.

Table 1: Research Findings and Existing Literature

S. No	Roll Number	Name	Paper Title	Tools / Tech	Findings	Citation
1	101803323	Nimish Mathur	Image colorization using GAN	GANs	Working of GANs	Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi

2	101803223	Shubham Goenka	High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs	Conditional GANS	Working of Conditional GAN	Ting-Chun Wang Ming- Yu Liu Jun-Yan Zhu Andrew Tao Jan Kautz Bryan Catanzaro
3	101803188	Jadi Akhilesh Prasad	Image-to-Image Translation with Conditional Adversarial Networks	GANs	Conditional GANs	Isola P., Zhu J., Zhou T., and Alexei A
4	101803294	Dev Kevlani	Gray-Scale Image Colorization Using Cycle-Consistent Generative Adversarial Networks with Residual Structure Enhancer	Grey Scale image color. Using cycle consistent GANS	Working of cycle consistent GANS	Mohammad Mahdi Johari, Hamid Behroozi
6	101803323	Nimish Mathur	Image colorization using GANS using transfer learning	Image color. Using transfer learning	Working of transfer learning in GANs	Leila Kiani Masoud Saeed Hossein Nezamabadi - pour
7	101803188	Jadi Akhilesh Prasad	Automatic Sketch Colorization using DCGAN	DCGAN	Working of DCGAN	Hwan Heo and Youngbae Hwang

### **2.1.4 The problem that has been identified**

It is very difficult to train GAN to generate high-resolution colored images from grayscale images. Simply adding more up sampling layers in state-of the-art GAN models for generating high-resolution images generally results in training instability and produces nonsensical outputs. We decompose the problem of grayscale to colored image synthesis through DCGAN by generalizing the colorization procedure to high resolution images and suggest training strategies that speed up the process and greatly stabilize it.

### **2.1.5 Survey of Tools & Technologies Used**

A few of the notable technologies used in the project are: -

1. Image Processing (Normalization)
2. Deep Learning:
  - a. Convolutional Neural Networks
  - b. Artificial Neural Networks
  - c. Batch Normalization
  - d. Dropout
  - e. Max pooling
  - f. Adam Optimization

## 2.2 Standards

IEEE standards are followed for complete development process of this software.

Software practices & respective codes followed in this process are shown in table 2.

Phase or activity group	Number	Standard bite
Requirement's specification	IEEE 830 IEEE 1233 IEEE1320	Recommended practice for software requirements specifications Guide for developing system requirements specifications Function modeling language
Design	IEEE 1016 IEEE 1471	Software design descriptions Recommended practice for architectural description of software-intensive systems
Implementation, acquisition and tools	IEEE 1062 IEEE 1462 IEEE 1175	Software acquisition Guideline for the evaluation and selection of CASE tools Guide for CASE tool interconnection
Testing	IEEE 829 IEEE 1008 IEEE 1012 IEEE1059 IEEE 1028 IEEE 1044	Software test documentation Software unit testing Software verification and validation Guide for software verification and validation plans Software reviews and audits Classification for software anomalies
Maintenance	IEEE 1219	Software maintenance

Table 2: Software Engineering Standards

SRS building process completely followed from IEEE guide to build a SRS (830-1998 - IEEE). CASE tools used for constructing different software diagrams. For code implementation purpose google coding standards for conventions & layout is use. For testing & its documentation IEEE standards will be used in as part of future scope of project.



## **2.3 Software Requirements Specification**

### **2.3.1 Introduction**

#### **2.3.1.1 Purpose**

The aim of this report is to present a thorough account of the software components which are going to be implemented on PyTorch/Keras machine learning frameworks aimed to generate appropriate image on the basis of image description given by the user.

#### **2.3.1.2 Intended Audience & Reading Suggestions**

Audience could include users like teachers, scientists, architects, interior and graphic designers. Every normal person could convert degraded and vintage photographs by specifying it to the DCGAN model in no time and will in turn get great quality images. This model will also benefit architects and interior designers as designing buildings and its interiors could easily be accomplished by providing input to the model. Also, filtering and editing images could easily be done thus benefitting the job of graphic designers.

#### **2.3.1.3 Project Scope**

At the current research point, the images are visually plausible only if the output is taken as a lower resolution. Higher resolution could create a whole new dimension in the use case. Given limited GPU memory and resources, the training can only be done on a smaller dataset focusing on a few topics at a time. The complexity of the situation could be handled once the resources are available in the coming years. The project has the ability to act as an imagination unit for an AI-powered humanoid.

### **2.3.2 Overall Description**

#### **2.3.2.1 Product Perspective**

The work aims to generate colored image from grayscale image. The result shows that generated images are consistent with input grayscale image. Recently, deep CNN and

RNN for image have yielded highly discriminative and generalizable image representations learned automatically from pixels and matrix. Motivated by these works, we aim on mapping directly from pixels and matrix to image pixels. Our main contribution in this work is to develop a simple and effective GAN architecture and training strategy that enables compelling grayscale image to colored image [15] synthesis from human input image.

### 2.3.2.2 Product Features

Firstly, the input data, i.e., a grayscale is preprocessed. It is important to extract meaning before they are fed to DCGAN. Basically, GANs consist of a generator and a discriminator that compete in a two-player minimax game. The Generator Network takes the input and tries to generate a sample of data. It then generates a data which is then fed into a discriminator network. The task of Discriminator Network is to take input either from the real data or from the generator and try to predict whether the input is real or generated. For the generator and discriminator models, we followed Deep Convolutional GANs (DCGAN) [17] guidelines and employed convolutional networks in both 5 generator and discriminator architectures. The architecture was also modified as a conditional GAN instead of a traditional DCGAN; we also follow guideline in [5] and provide noise only in the form of dropout [18], applied on several layers of our generator. The architecture of generator  $G$  is the same as the baseline. For discriminator  $D$ , we use similar architecture as the baseline's contractive path: a series of  $4 \times 4$  convolutional layers with stride 2 with the number of channels being doubled after each down sampling. All convolution layers are followed by batch normalization, leaky ReLU activation with slope 0.2. After the last layer, a convolution is applied to map to a 1-dimensional output, followed by a sigmoid function to return a probability value of the input being real or fake. The input of the discriminator is a colored image either coming from the generator or true labels, concatenated with the grayscale image.

### **2.3.3 External Interface Requirements**

The primary input to the system is an image by the user. The image should be in accordance with the image dataset on which the model is trained. After entering the grayscale image, user also inputs the resolution of the image required to be generated from the given options. Software is also open to user input in terms of his own dataset, i.e., user can train his own dataset. User can model his own dataset by providing the set of images and preprocessed grayscale image, i.e., image tensors. Then the software, using the image dataset and the trained model, generates the required image. The image is in accordance with the image provided.

#### **2.3.3.1 User Interfaces**

A graphical user interface is created which provides user-friendly environment for generating artistic images. The user inputs the text into the text field provided the text should comply with image dataset used for training. The interface links to the modules required for converting the given input text into text embedding and consequently, into the desired image through a series of up sampling and down sampling blocks. The output image is then shown on the screen.

#### **2.3.3.2 Hardware Interfaces**

Not Applicable.

#### **2.3.3.3 Software Interfaces**

In this system, there is an API which acts as a messenger that delivers the request, i.e., the input text in our case, to the DCGAN model and then delivers the desired image back to the user by displaying it on screen using user interface. Its acts like a software intermediary that allows the user to interact with the model.

## **2.3.4 Requirements**

### **2.3.4.1 Functional Requirements**

1. Training Model: Parameters are trained by automatically extracting features from the image.
2. Image analysis: Image training data is analyzed and preprocessed.
3. Synthesize image: Pass the training set into the Neural Network to get the desired output.

### **2.3.4.2 Non-Functional Requirements**

1. Performance- the action or process of performing a task or function
2. Scalability- the ability of a computing process to be used or produced in a range of capabilities.
3. Availability- the presence and free nature of the services provided
4. Recoverability- To get back, especially by making an effort
5. Maintainability- probability of performing a successful repair action within time.
6. Serviceability- capable of or being of service
7. Security- in terms of protecting the authenticity of the user
8. Regulatory- serving or intended to regulate something
9. Interoperability- the operating efficiency between different interfaces, the ability of computer systems or software to exchange and make use of information
10. Data integrity- preservation of, guarantee of the accurateness and constancy of data on whole life-cycle

### **2.3.4.3 Performance Requirements**

The performance of our system is measured through how good the images are generated from the input. If the image is visually recognizable in reference to the input entered and up to the standards of today's quality of images then only will the performance requirements will be met.

#### **2.3.4.3.1 Simplicity**

For the user interface, the main thing to keep in mind is to keep it simple and easy to use. It should not be uselessly sophisticated and complicated and usable for a layman. The user should only give input and get the result.

#### **2.3.4.3.2 Availability**

The interface will be in use and accessible to the user till the PC or laptop or the device is switched on and running.

#### **2.3.4.3.3 Maintainability**

Our system is specified for the task of generating lifelike images given the text description and henceforth rough doodles. Therefore, maintainability does not have more importance than it has in a normal software.

## **2.4 Risk Analysis**

Since the audience involved for the software are mostly indulged with sensitive tasks such as teaching students etc., there is always a risk of generating images which are not the best approximation of the images in raw form and hence, this could lead to barriers for some particular tasks until the model gives totally perfectionist results.

## METHODOLOGY ADOPTED

---

### 3.1 Investigative Techniques

The investigative techniques employed by the team members are shown in table 3.

Table 3: Investigative Techniques

S. No.	Investigative Projects Techniques	Investigative Projects Description
1	Descriptive	<ul style="list-style-type: none"><li>• Research at conceptual level on GANs, DCGANs, Conditional GANs.</li><li>• Creating an optimal architecture of CNN to generate GANs.</li></ul>
2	Comparative	<ul style="list-style-type: none"><li>• Comparing different extensions of GANs to generate the best possible output for text to image conversion.</li><li>• Comparing different possible architectures and tuning the hyper-parameters.</li></ul>
3	Experimental	<ul style="list-style-type: none"><li>• Using machine learning/deep learning concepts on different frameworks, i.e. Keras, PyTorch.</li></ul>

### 3.2 Proposed Solution

Image colorization is an image-to-image translation problem that maps a high dimensional input to a high dimensional output. It can be seen as pixel-wise regression problem where structure in the input is highly aligned with structure in the output. That means the network needs not only to generate an output with the same spatial dimension as the input, but also to provide color information to each pixel in the grayscale input image. We provide an entirely convolutional model architecture using a regression loss as our baseline and then extend the idea to adversarial nets. In this work we utilize the

L\*a\*b\* color space for the colorization task. This is because L\*a\*b\* color space contains dedicated channel to depict the brightness of the image and the color information is fully encoded in the remaining two channels. As a result, this prevents any sudden variations in both color and brightness through small perturbations in intensity values that are experienced through RGB.

### 3.3 Work Breakdown Structure of image colorization using GANs

The work breakdown structure of image colorization using GANs is shown in Figure 2.

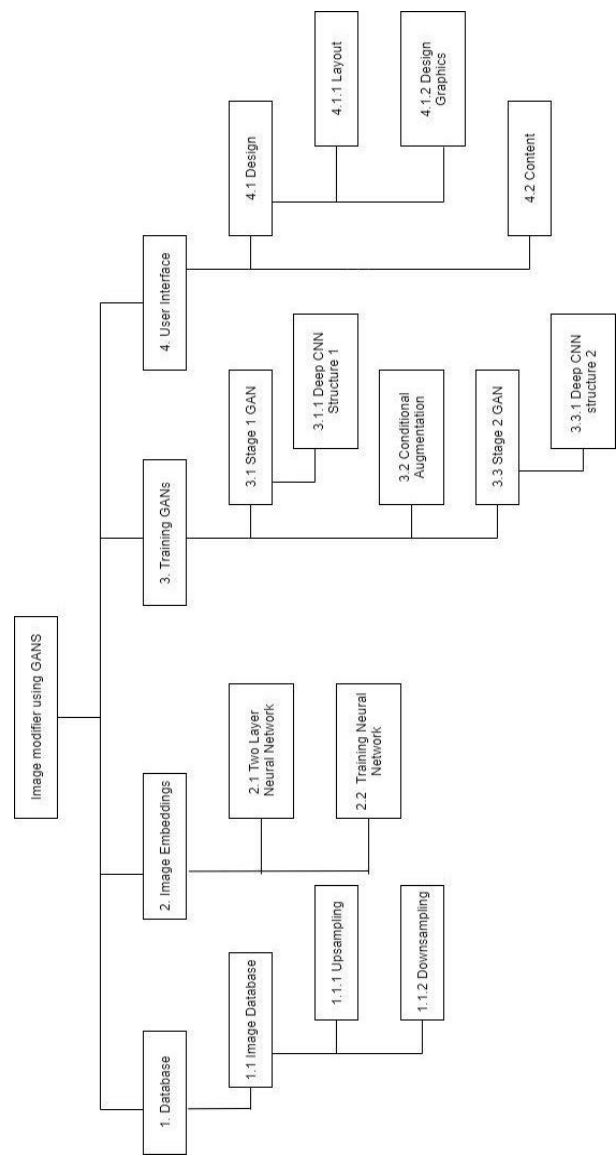


Figure 2: Work Breakdown Structure of Image Colorization using GANs

### **3.4 Tools & Technologies Used Technologies**

- Artificial Neural Networks
- Convolutional Neural Networks
- Generative Adversarial Networks
- Parallel Training
- Batch Normalization
- Dropout

#### **Tools**

- TensorFlow
- PyTorch
- CUDA
- Google Collab
- NumPy
- SciPy



### 4.1 System Architecture

#### 4.1.1 MVC Architecture

The MVC architecture of the proposed project is shown in Figure 3.

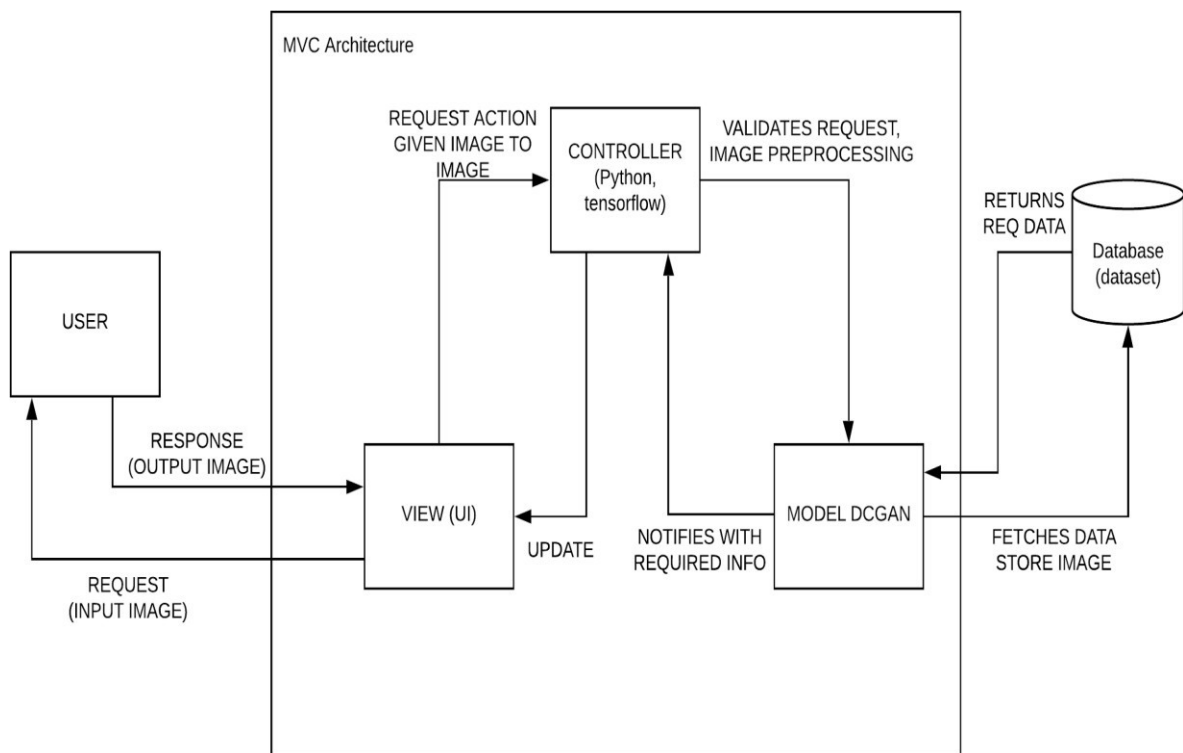


Figure 3: MVC Architecture

### 4.1.2 3 Tier Diagram

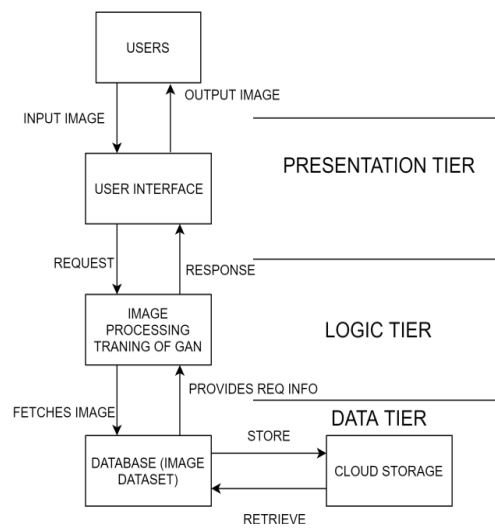


Figure 4: 3 tier

### 4.1.3 User Interface Diagram

The user interface diagram for the model is shown in Figure 5.

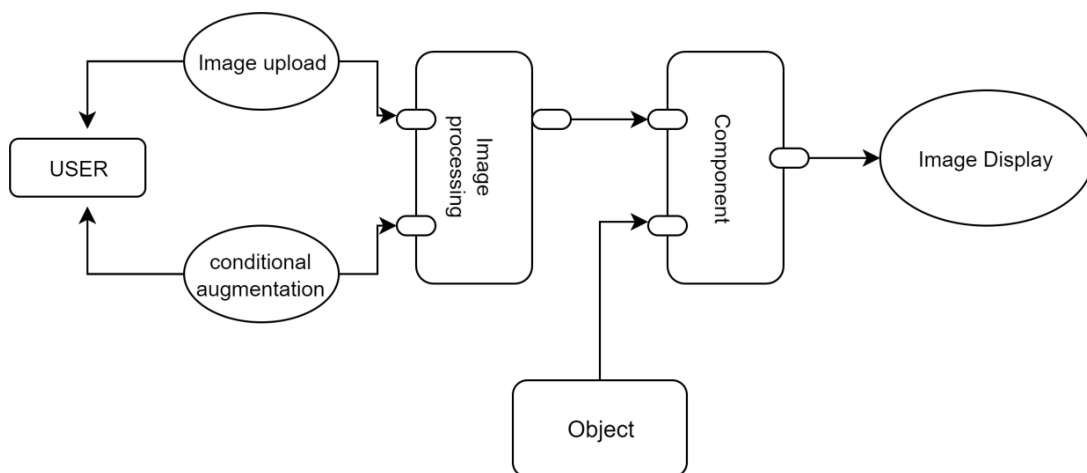


Figure 5: User Interface Diagram

#### 4.1.4 Data Model

The data model diagram for the system is shown in Figure 6.

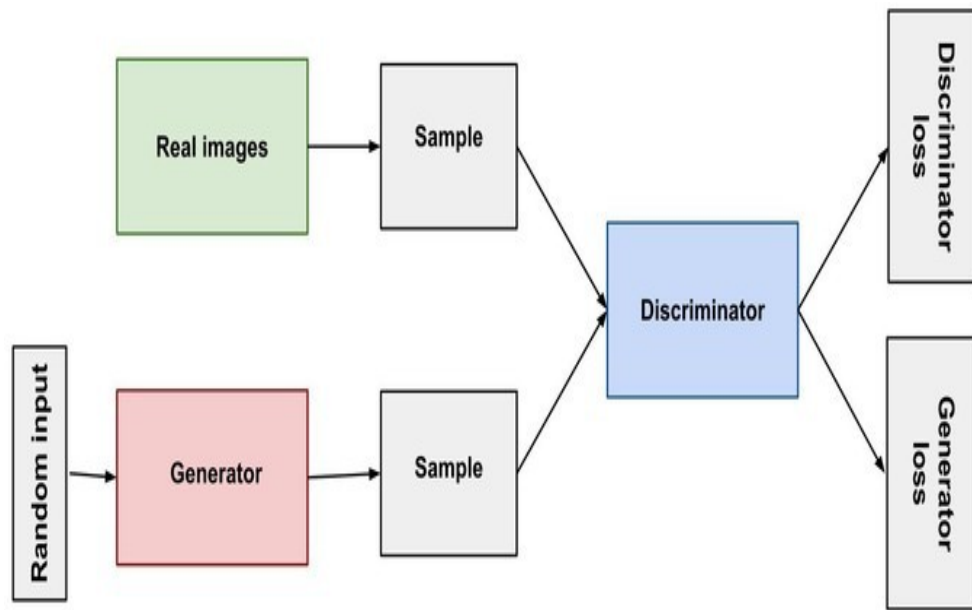


Figure 6: Data Model diagram

## 4.2 Design Level Diagrams

### 4.2.1 Flowchart

The class diagram drawn for the project is shown in Figure 7.

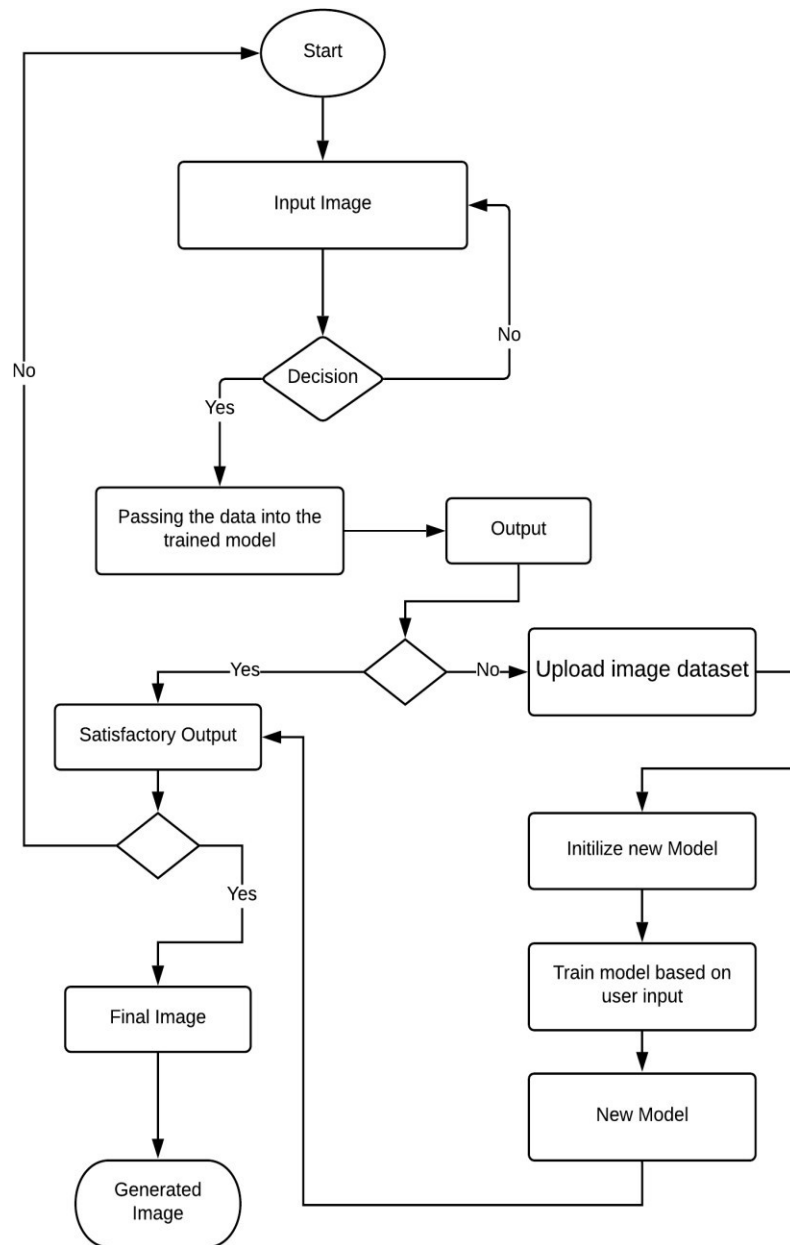


Figure 7: Flowchart

## 4.2.2 Sequence Diagram

The Sequence Diagram for the proposed project is shown below.

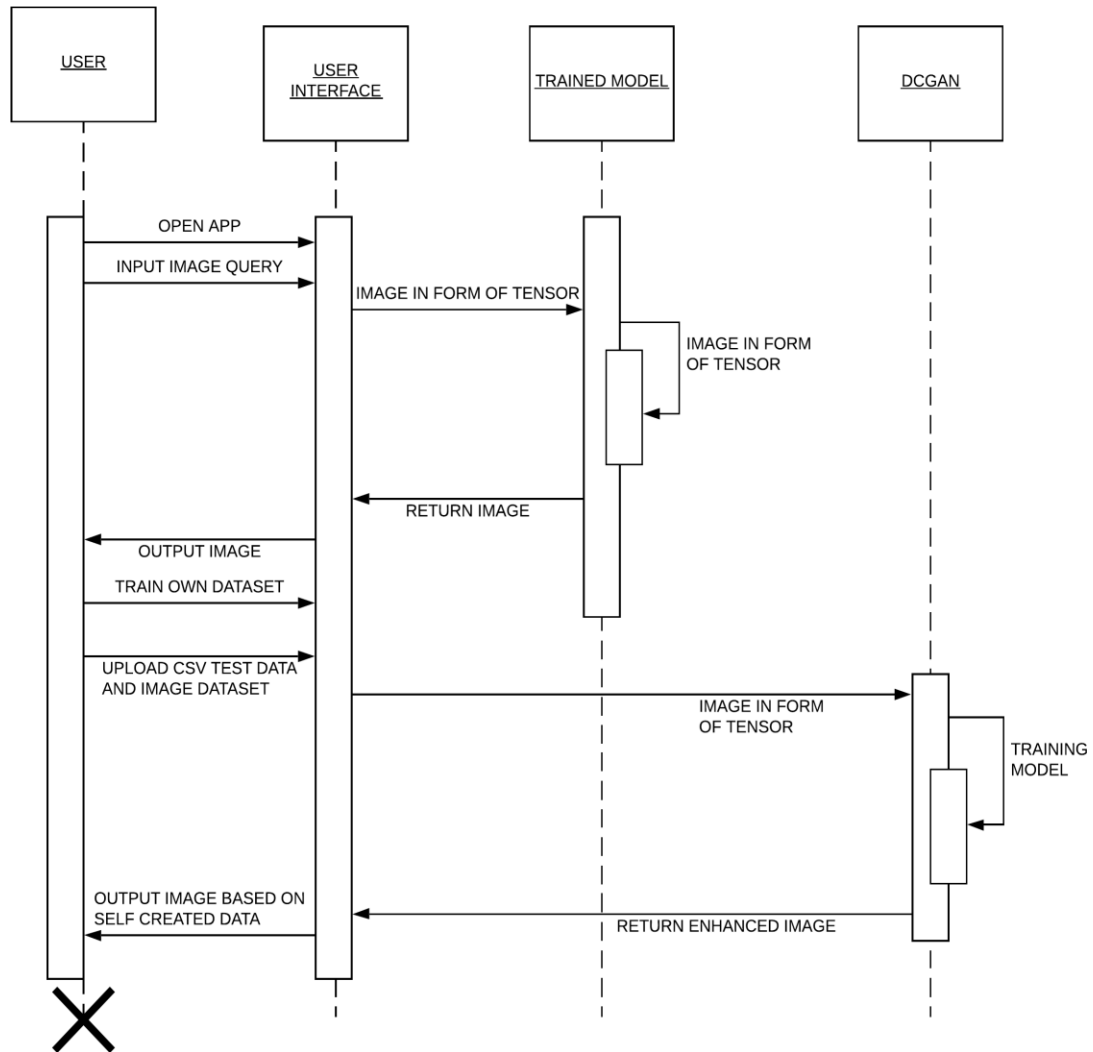


Figure 8: Sequence Diagram

### 4.2.3 Entity-Relationship Diagram

The entity-relationship diagram drawn for the project is shown in Figure 9.

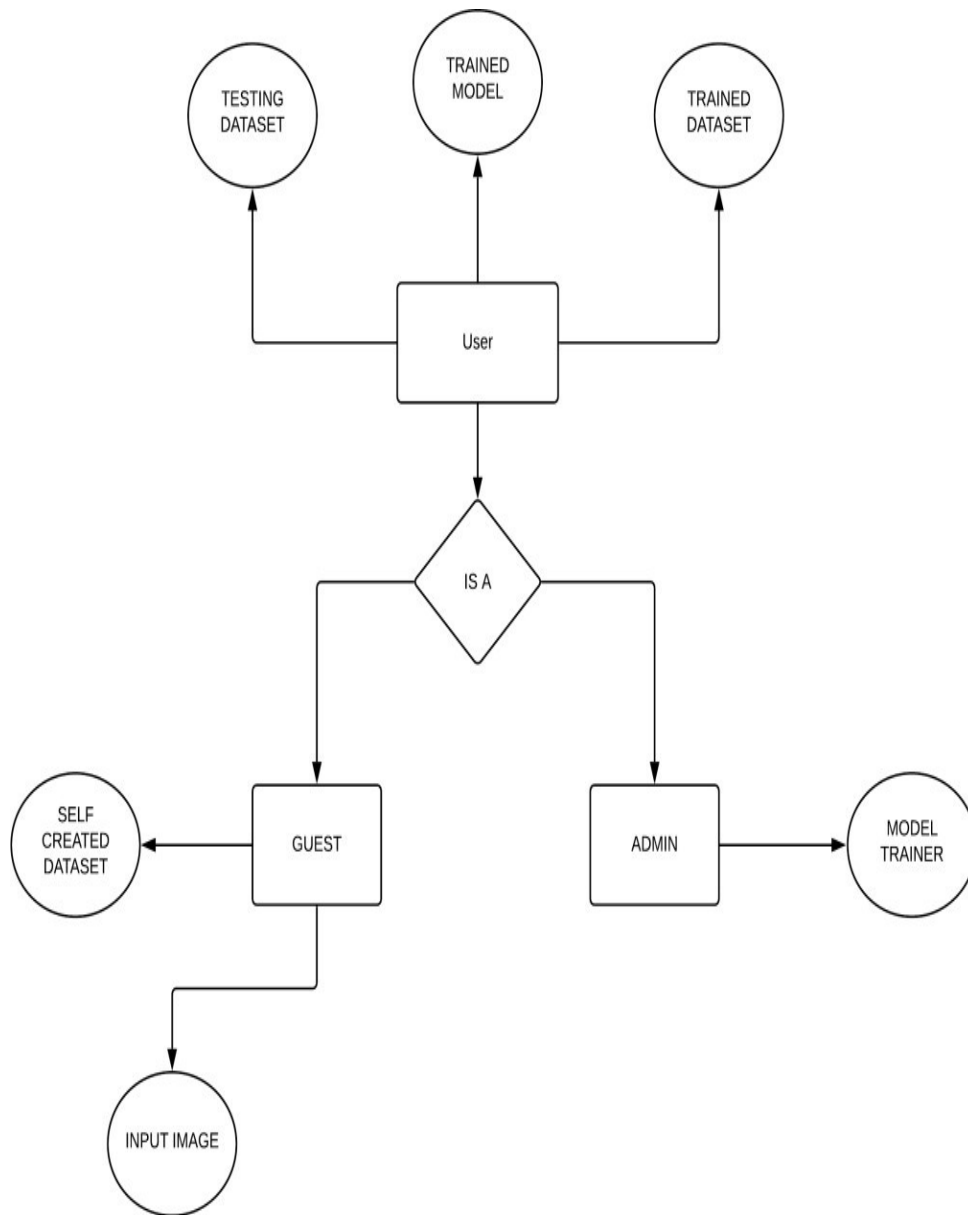


Figure 9: ER Diagram

### 4.3 User Interface Diagrams

The user interface diagram for the proposed project is shown in Figure below with use case templates

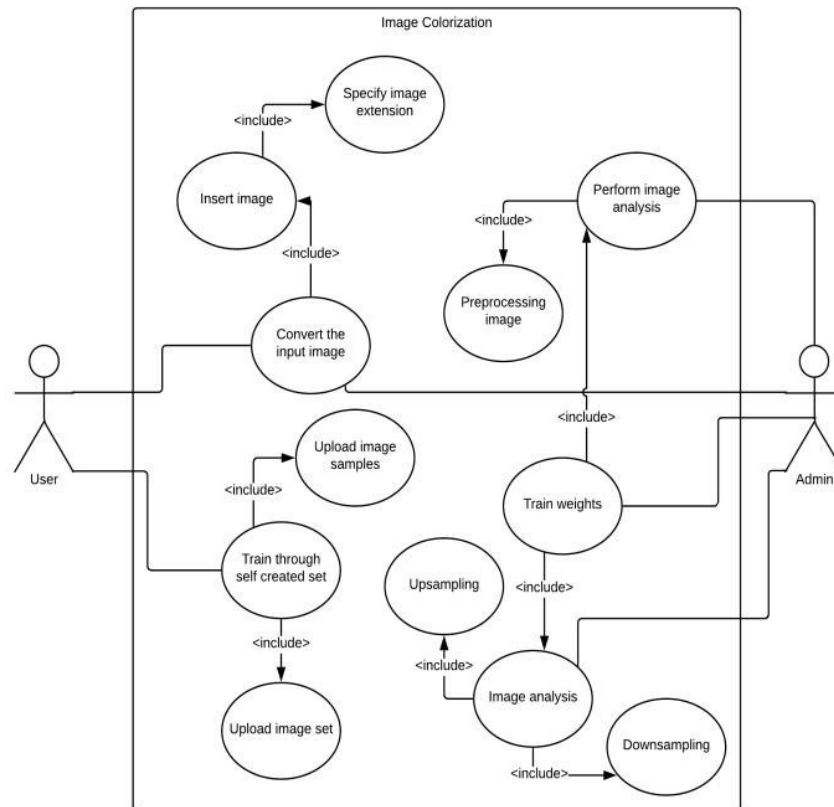


Figure 10: Use Case Diagram

### Use-Case Templates

ID:	UC-1
Title:	Convert the input image
Description:	Passes the input image data into the DCGAN structure to form the desired output image.
Actor:	Client

Precondition:	Client is logged into system
Postcondition:	The output image is saved into the local directory.
ID:	UC-2
Title:	Insert image
Description:	Client accesses the system and inputs image as per the model already trained.
Actor:	Client
Precondition:	Client is logged into system
Postcondition:	Image will be converted to its corresponding tensor when passed through the trained model.

ID:	UC-3
Title:	Specify image extension
Description:	Client accesses the system and provides the image extension he wishes to display the image as.
Actor:	Client
Precondition:	Client is logged into system
Postcondition:	Image in the specified format (jpeg,png,jpg) is saved in the local directory.

ID:	UC-4
Title:	Perform image analysis
Description:	Training data is analyzed and preprocessed.
Actor:	Admin
Precondition:	The tensor form of images are available.
Postcondition:	Trained model.



ID:	UC-5
Title:	Preprocessing image
Description:	It transforms the image in the desired format.
Actor:	Admin
Precondition:	The training images are in pre-decided format.
Postcondition:	Images in desired format

ID:	UC-6
Title:	Train through self-created set
Description:	The user can train the deep neural network structure using his/her own image samples for more specific and efficient image generation .
Actor:	Client
Precondition:	Client logged into the system.
Postcondition:	The model will be trained.

ID:	UC-7
Title:	Upload image samples
Description:	Image samples are uploaded in order to train the model.
Actor:	Client
Precondition:	Samples in correspondence to the desired category of images should be present in local directory.
Postcondition:	Batch of training set ready to input into the model.

ID:	UC-8
Title:	Upload image zip file
Description:	Images in zip file are uploaded in order to train the model.

Actor:	Client
Precondition:	Images corresponding to the samples should be zipped in a file before uploading.
Postcondition:	Zip file is extracted to form a training set.

ID:	UC-9
Title:	Image analysis
Description:	Image training data is analyzed and preprocessed.
Actor:	Admin
Precondition:	Images specified to the subject should be ready in the local directory.
Postcondition:	Image data is converted to pixel formats.

ID:	UC-10
Title:	Up sampling
Description:	The image is up sampled by using a Convolutional Neural Network appended to the GAN structure.
Actor:	Admin
Precondition:	Image should be available
Postcondition:	Up sampled image with better resolution is formed.

ID:	UC-11
Title:	Down sampling
Description:	The image is down sampled by using a Convolutional Neural Network before feeding into the GAN structure.
Actor:	Admin
Precondition:	Image should be available

Postcondition:	A down sampled image with features good enough to be extracted is formed.
ID:	UC-12
Title:	Train weights
Description:	Parameters are trained by automatically extracting features from the image.
Actor:	Admin
Precondition:	Preprocessed image samples.
Postcondition:	Trained model.

# IMPLEMENTATION AND EXPERIMENTAL RESULTS

---

## 5.1 Experimental Setup

For the model “CIFAR-10” dataset was used which consisted of 60000 32x32 color images in 10 classes, with 6000 images per class.

The dataset used for the final experimentation is “CIFAR-10” We used DCGAN architecture to work on this dataset[8]Google Collab(is a free cloud service and now it supports free GPU and develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV)by building our architecture using PyTorch.

### 5.1.1 Data (Birds Dataset)

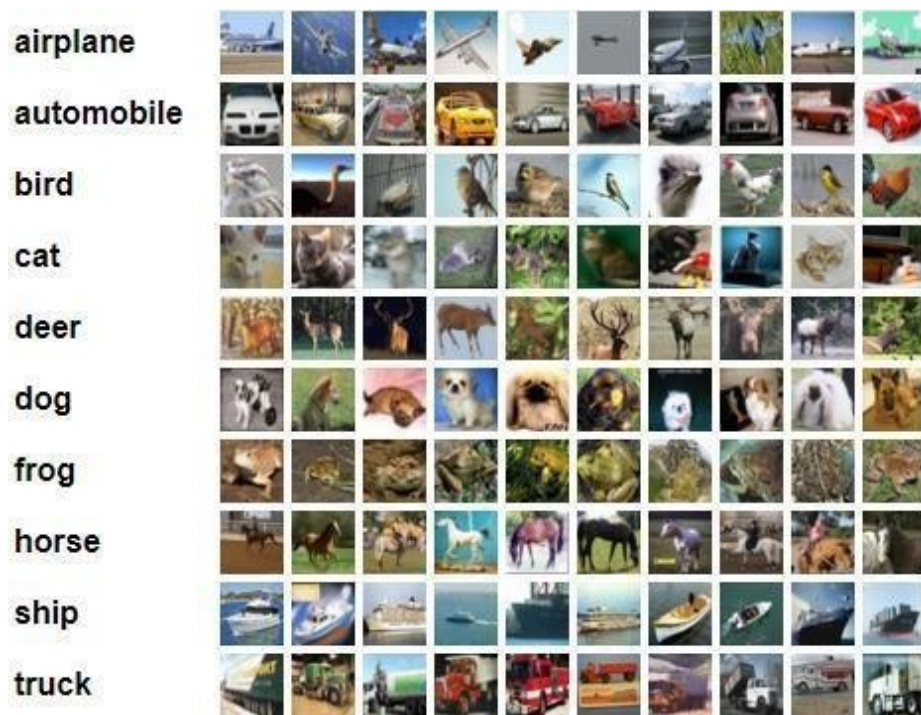


Figure 11: CIFAR-10 Dataset

The CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

### 5.1.2 Performance Parameters

To measure the performance, we have chosen to employ mean absolute error (MAE) and accuracy. MAE is computed by taking the mean of the absolute error of the generated and source images on a pixel level for each color channel. Accuracy is measured by the ratio between the number of pixels that have the same color information as the source and the total number of pixels. Any two pixels are considered to have the same color if their underlying color channels lie within some threshold distance  $\epsilon$ . This is mathematically represented by

$$acc(x, y) = \frac{1}{n} \sum_{p=1}^n \prod_{\ell=1}^3 \mathbf{1}_{[0, \epsilon_\ell]}(|h(x)^{(p, \ell)} - y^{(p, \ell)}|)$$

where  $\mathbf{1}_{[0, \epsilon]}(x)$ ,  $x \in \mathbb{R}$  denotes the indicator function,  $y$  is the corresponding color image,  $h$  is a function mapping from grayscale to color images, and  $\epsilon_\ell$  is a threshold distance used for each color channel

## 5.2 Working of Project

### 5.2.1 Procedural Workflow

For our baseline model, we follow the "Fully Conventional Network" [6] model, where fully connected layers are replaced by convolutional layers, including up sampling instead of pooling operators. The idea is based on the encoder decoder network [7], where the input is progressively reduced by using a series of compressive encoding layers, and then reversing the process using a series of extender decoding layers to reorder the input are

given. Using this method we can train the model end-to-end without consuming large amounts of memory. Note that the subsequent downsampling leads to a much more compact feature learning in the middle layers. This strategy constitutes an important feature for the network, otherwise the resolution will be limited by GPU memory. The baseline model needs to find a direct mapping from the grayscale imagespace to the color image space. However, there is an information bottleneck that prevents the flow of low-level information in the network in the encoder-decoder architecture. To fix this problem, features are eliminated by the contracted path in a wide path within the network. It shares inputs and outputs the locations of key edges in grayscale and color images. This architecture is called U-NET [10], where skip connections are added between layer I and layer N-I. The architecture of the model is symmetric, with n encoding units and n decoding units. The contraction path consists of  $4 \times 4$  convolution layers, with strass 2 for downsampling, each followed by batch normalization [9] and leak-rail[19] Activation action with a slope of 0.2. The number of channels doubles after each stage. Each unit in the paved path has a  $4 \times 4$  transposed conventional layer with 2 for stamping, a contraction with an activation map of the mirroring layer in the contracting path, followed by a batch normalization and reboot activation function. The last layer of the network is a  $1 \times 1$  convolution which is equivalent to the cross-channel parametric pooling layer. We use the Tan function for the last layer, as proposed by [5]. The number of channels in the output layer is 3 with  $L * a * b * \text{color space}$ .

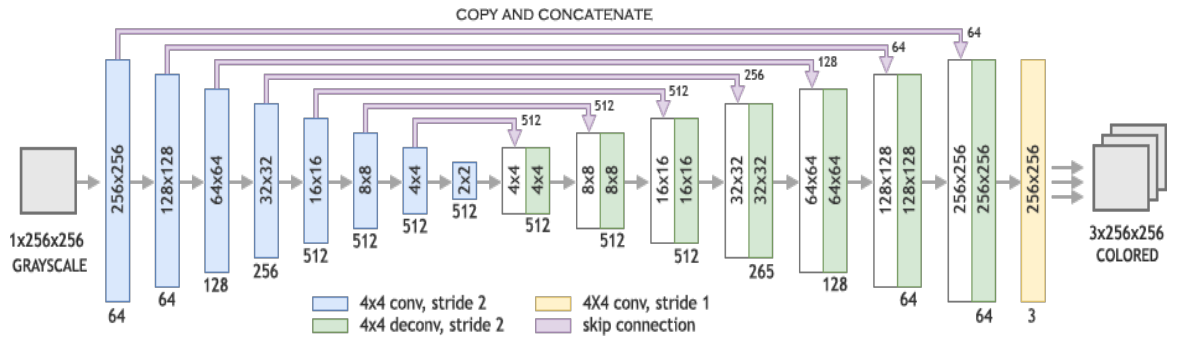


Figure 12: UNET Architecture(256\*256)

We train the baseline model to minimize the Euclidean distance between predicted and ground truth averaged over all pixels:

$$J(x; \theta) = \frac{1}{3n} \sum_{\ell=1}^3 \sum_{p=1}^n \|h(x; \theta)^{(p, \ell)} - y^{(p, \ell)}\|_2^2$$

where  $x$  is our grayscale input image,  $y$  is the corresponding color image,  $p$  and  $\ell$  are indices of pixels and color channels respectively,  $n$  is the total number of pixels, and  $h$  is a function mapping from grayscale to color images.

### 5.2.2 Algorithmic Approaches Used

For the generator and descriptor model, we followed the Deep Contextual Gains (DCGAN) [17] guidelines and employed convolution networks in both generator and descriptor architectures. The architecture was also modified to be a conditional GAN instead of a traditional DCGAN; We also follow the guidelines in [5] and only make noise in the form of dropouts [18] on many layers of our generator. The architecture of generator  $G$  is similar to the baseline. For Discriminator  $D$ , we use the same architecture as the baseline contractive path: a series of  $4 \times 4$  convolutional layers with 2 strides doubling the number of channels after each downsampling. All the convolution layer is followed by the normalization, leaky ReLU activation of the batch with slope 0.2. After the last layer, a convolution is applied for mapping to 1 dimensional output, followed by a sigmoid function to return the input or probability value to be real. The input of the Discriminator is a color image that comes from either the generator or the True Label, which is included with the grayscale image.

To train our network, we used Adam [15] optimization and the introduction of weights as proposed by [8]. We used an initial learning rate of  $2 \times 10^{-4}$  for both generator and discrepant and manually decayed the learning rate by a factor of 10 whenever the loss function began to plateau. For the hyper-parameter  $\lambda$  we followed the protocol from [5] and chose  $\lambda = 100$ , which forces the generator to produce a picture similar to the ground truth. Ganes are considered very difficult to train because it requires a Nash equilibrium of non-convex games with continuous, high dimensional parameters [13]. We followed a set of constraints and techniques proposed by [5,13,1], 1 encourage] to encourage the convergence of our firm GAN and train it to stabilize.

**Alternative Cost Function:** This genetic alternative cost function [20] was chosen due to its redundant nature; The motivation for this cost function is to ensure that each player has a strong shield when that player is "losing" the game..

**One Sided Label Smoothing:** Deep neural networks typically produce highly confident outputs when used in classification. It has been shown that replacing a target of 0 and 1 for a classifier with smooth values such as 1 and .9 is a regular routine for a comfortable network [12]. Salimans et al [17] demonstrated that smoothing one-way labels would help the discriminator to infer softening probabilities and reduce the vulnerability of GANs to adversarial instances. In this technique we only smooth positive labels to 0.9, leaving negative labels set to 0.

**Batch Normalization:** One of the main difficulties for training GANs is to collapse a parameter setting for a generator where it always emits the same output [17]. This phenomenon is called mode-collapse, also known as Helvetica scenario [20]. When mode-collapse has occurred, the generator learns that a single output is capable of continuously dodging the differential. This is non-ideal because the goal is to learn the distribution of the network and not the 6 most ideal ways to fool the discriminator. Batch normalization [9] proves necessary to train both networks, preventing both samples from destroying all samples at the same point [17]. The batch-norm is not applicable to the first layer of the generator and the descriptor and the last layer of the generator as suggested by [5].

**All Convolutional Net:** Stranded convolutions are used instead of spatial pooling functions. This effectively allows the model to learn its own downsampling / upsampling rather than relying on a certain downsampling / weathering method. This idea was proposed in [15] and has been shown to improve training performance, as the network learns all the required attacks with only firm layers.

**LeakyReLU Activation Function:** Radford et al. [13] has been shown that using leaky



ReLU [5] activation functions in the Discriminator results in better performance when using regular ReLU. We also found that using leaky ReLU in the encoder part of the generator suggested by [21] works a little better.

### 5.2.3 Project Deployment

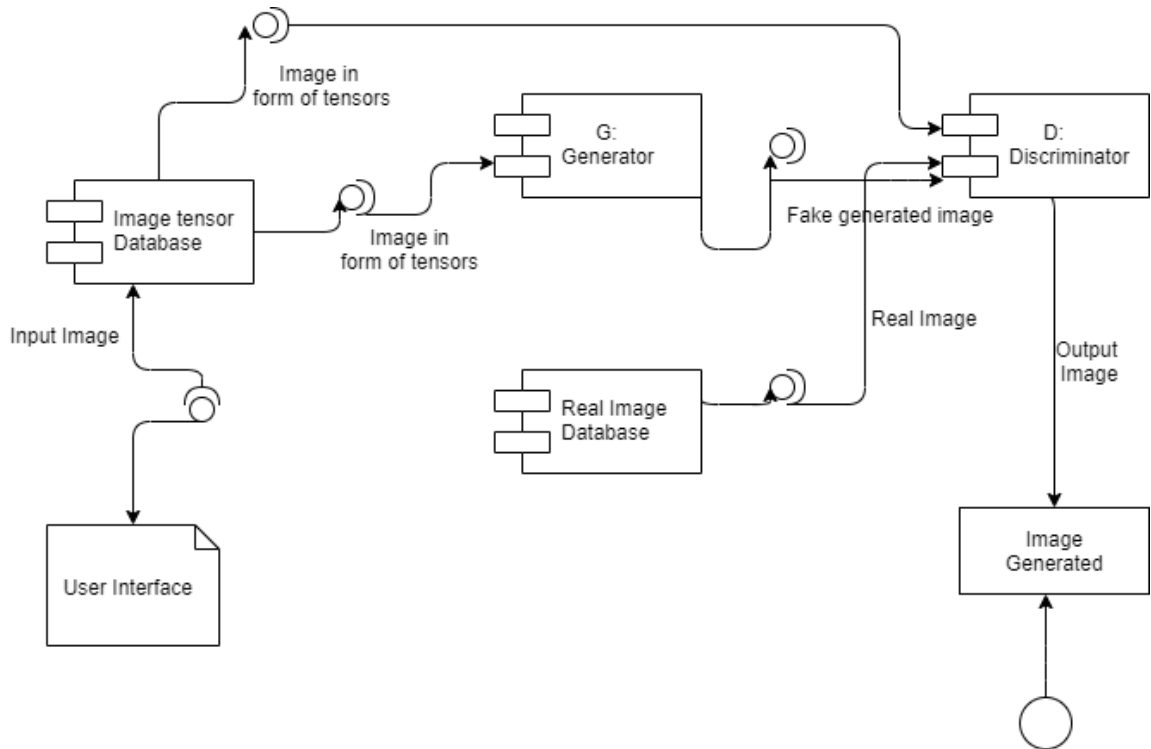


Figure 13: System Component Diagram

## 5.2.4 System Screenshots

```
gan_colorization.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
RAM Disk
Press F11 to exit full screen
+ Code + Text Copy to Drive
Files
<>
data
drive
grey_data
sample_data
print('Finished Training')
...
[1, 331] d_loss: 1.22194 g_loss: 7.81389
[1, 341] d_loss: 1.21620 g_loss: 8.42258
[1, 351] d_loss: 1.25866 g_loss: 7.56594
[1, 361] d_loss: 1.22151 g_loss: 8.16087
[1, 371] d_loss: 1.21336 g_loss: 7.83002
[1, 381] d_loss: 1.29598 g_loss: 8.26566
[1, 391] d_loss: 1.19013 g_loss: 8.34061
[2, 1] d_loss: 0.16313 g_loss: 0.83587
[2, 11] d_loss: 1.13369 g_loss: 8.41480
[2, 21] d_loss: 1.30896 g_loss: 8.55154
[2, 31] d_loss: 1.28211 g_loss: 8.18660
[2, 41] d_loss: 1.23489 g_loss: 7.99472
[2, 51] d_loss: 1.19348 g_loss: 8.27063
[2, 61] d_loss: 1.28191 g_loss: 7.88005
[2, 71] d_loss: 1.30734 g_loss: 8.23262
[2, 81] d_loss: 1.25528 g_loss: 8.21677
[2, 91] d_loss: 1.25108 g_loss: 7.89757
[2, 101] d_loss: 1.16258 g_loss: 8.22964
[2, 111] d_loss: 1.33981 g_loss: 8.14641
[2, 121] d_loss: 1.38948 g_loss: 8.04133
[2, 131] d_loss: 1.22540 g_loss: 8.14649
[2, 141] d_loss: 1.25181 g_loss: 7.86047
[2, 151] d_loss: 1.26457 g_loss: 8.01449
[2, 161] d_loss: 1.30177 g_loss: 8.15612
[2, 171] d_loss: 1.14059 g_loss: 8.44266
[2, 181] d_loss: 1.35748 g_loss: 7.68392
[2, 191] d_loss: 1.15068 g_loss: 7.87364
[2, 201] d_loss: 1.36726 g_loss: 7.81587
[2, 211] d_loss: 1.22743 g_loss: 7.94644
[2, 221] d_loss: 1.39004 g_loss: 8.18650
[2, 231] d_loss: 1.18701 g_loss: 7.95168
[2, 241] d_loss: 1.19927 g_loss: 8.37821
[2, 251] d_loss: 1.37754 g_loss: 7.84287
[2, 261] d_loss: 1.28562 g_loss: 7.97987
[2, 271] d_loss: 1.15162 g_loss: 8.45784
[2, 281] d_loss: 1.19668 g_loss: 8.48829
[2, 291] d_loss: 1.30190 g_loss: 8.30408
[2, 301] d_loss: 1.23634 g_loss: 8.37869
[2, 311] d_loss: 1.26305 g_loss: 8.29427
Disk 35.57 GB available
Executing (24m 44s) Cell
```

Figure 14: Training DCGAN



Figure 15: Fake Images Generated



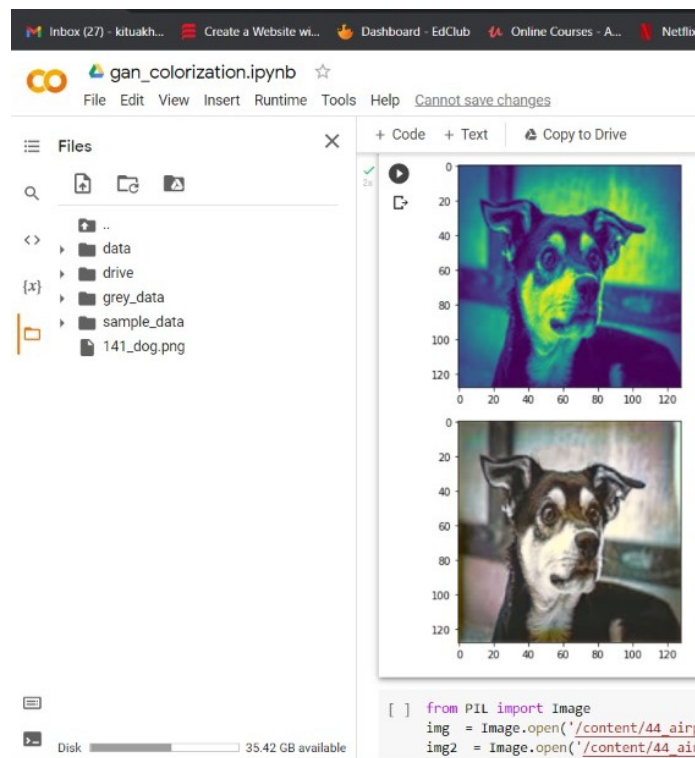
Figure 16: Real Images



Figure 17:Fake Images(Batch-8)



Figure 18: Real Images(Batch-8)



## **5.3 Testing Process**

This section describes the complete testing process including the techniques employed for testing and fixing bugs, test cases and the results after testing.

### **5.3.1 Test Plan**

The main idea behind the test plan consists of testing our proposed model with different hyper-parameters. The testing of the model was done on the test dataset of CIFAR-10.

#### **5.3.1.1 Features to be tested**

- Quality
- Structure of core entity
- Background perfection
- Pixel density

### 5.3.1.2 Test Strategy and Techniques

The purpose of this module is to receive text descriptions and use the already trained model to generate new images. The testing is done on the basis of Quality, Structure of core entity, Background perfection and Pixel Density of the generated image.

### 5.3.2 Test Cases

The following were the important test cases to properly gauge the working of the software:

- Performance: Performance was tested under different hyper parameters. Overfitting and under fitting were removed during the process.
- Outputs under different circumstances: The model was tested under different circumstances with images under different environments.
- Dataset Variances: Testing of the model using dataset CIFAR-10 was done which has 10 different labels (cats,frogs,airplanes,cars,bird,deer dog,horse,ship,truck).
- Model Architecture: Different architecture of GANs were implemented and the best one was found after testing under different datasets.

### 5.3.3 Test Results

- After repeated efforts in designing and implementing Generative Adversarial Networks, a final optimized model was created with minimal possible overfitting and under fitting.
- Multiple images with different environmental backgrounds and actions were formed. The images created could be identified to be changing correctly according to the images(tensors) passed.
- In some cases, the GAN was able to nearly replicate the ground truth. However, one drawback was that the GAN tends to colorize objects in colors that are most frequently seen. For example, many car images were colored red. This is most likely due to the significantly larger number of images with red cars than images with cars of another color

## 5.4 Results and Discussions

- The GPU integration for the model was working correctly at the training phase as well as testing phase.
- The images formed when passing the embedding had the correct quality, structure of core entity, background and pixel density.
- The DCGAN version clearly distinguished the background from the main entity hence giving improved quality and better pixel density.

## 5.5 Inferences Drawn

- Different extensions to the dataset can be added to get varied functionality from the project.
- Some of the outputs of the model are brilliant but many of them still aren't the most satisfying.
- GANs rely largely on the dataset, hence the dataset must be specifically arranged to fulfill the required functionality.
- To get the best functionality out of the model, we might need a huge dataset to cover all the entities in the world. But this seems quite unrealistic at the moment since GANs require high computations during training and the current hardware would probably take impractical time to train such model

## 5.6 Validation of Objectives Table

The validation of the proposed objectives is described in Table 4.

Table 4: Objectives Validation Table

S.No.	Objectives	Status
1	Training a conditional generative adversarial network to generate realistic colored images from greyscale images. The result will show that generated images are consistent with input image.	Yes
2	Lack of colorization techniques for black and white	Yes
	images could be handled.	
3	Generation of datasets through the model will be possible.	Yes



# CONCLUSIONS AND FUTURE WORK DIRECTION

---

## 6.1 Conclusions

With this project in progress deep learning principles are applied onto real life datasets and quantified results are obtained. We proposed a model which uses a simple DCGAN to generate colored images using a greyscale dataset. With the CIFAR-10 dataset, the model was able to consistently produce better looking (qualitatively) images than U-Net. Many of the images generated by U-Net had a brownish hue in the results known as the “Sepia effect” across  $L^*a^*b^*$  color space. This is due to the L2 loss function that was applied to the baseline CNN, which is known to cause a blurring effect.

## 6.2 Environmental, Economic and Societal Benefits

This project offers many benefits to its users such as:

- Restoration of old ,vintage and damaged pictures/black and white film reel
- This model will also benefit architects and interior designers as designing buildings and its interiors could easily be accomplished by providing grayscale model image input to the model.
- Model could be extended to videos converting black and white films to colored film.
- Generation of new dataset.

## 6.3 Reflections

We were able to study about all the basic principle about deep learning models i.e. their parameters, hyper-parameters and their tuning. We also understood that structuring and making modules for each utility works great and gives a robust experience. Planning work in advance using UML diagrams kept us focused towards our goal. Learning about TensorFlow, Keras etc. and the research involved during the process helped us stay updated about various state of the art AI solutions.

## **6.4 Future Work Plan**

Training the model on a much better machine with bigger dataset of high quality images. We would also need to seek a better quantitative metric to measure performance. This is because all evaluations of image quality were qualitative in our tests. Thus, having a new or existing quantitative metric such as peak signal-to-noise ratio (PSNR) and root mean square error (RMSE) will enable a much more robust process of quantifying performance.

### 7.1 Challenges faced

In this project, we generate colored images from greyscale images using GANs. As training through GANs results in not much high resolution images, we decided to train the dataset on a conditional GAN since the input was the greyscale image and no noise is needed as input so it had to be kept zero.

Training of GANs: Training through GANs also require high GPU power as it takes a lot of time to train the image and text dataset. We lack a powerful resource/ Machine to train the GANs as it required huge dataset and large number of iterations of the dataset to produce satisfactory results.

Compatibility Issues: Tensorflow/PyTorch compatibility issues were faced as they was not compatible with the windows.

Communication between team members: As our team members belong to different cities so we faced the challenge of communication gap during our summer vacation break. Our project progress was also halted during college placement drive. Working on the architecture of GANs for all team members simultaneously was not possible as we have only one machine with enough GPU computing power.

Working on new techniques: Due to many concepts of deep learning used, members had to do much research on deep learning in order to apply those in the model. Our project is to generate colored images from greyscale images using GANs and the problem faced in using GANs is that they are highly unpredictable and creating high resolution images are very challenging. Designing the optimal architecture of GANs is largely strenuous due to tuning of vast hyper parameters.

## 7.2 Relevant Subjects

The project embeds the following subjects as shown in Table 5.

Table 5: Relevant Subjects

Subject Code	Subject Name	Description
UML501	Machine Learning	<ul style="list-style-type: none"><li>• Basics of Python</li><li>• Neural Networks</li><li>• Convolutional Neural Networks</li></ul>
UCS503	Software Engineering	<ul style="list-style-type: none"><li>• Software Requirement Specification</li><li>• UML Diagrams</li><li>• Planning</li></ul>

### 7.3 Interdisciplinary Knowledge Sharing

The project requires the understanding of concepts like image normalization in which different types of images are normalized to the required specifications.

Concepts of Software Engineering which include all the planning and laying out the plan with the help of different tables, diagrams and templates (Software Requirement Specification, UML Diagrams, Use Case templates etc) helped in the process of completion of project

The other concepts used are of deep learning which includes convolution neural network. CNN or ConvNet is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing.

First we input image from our dataset into convolution layer and choose different parameters, apply filters with strides and padding according to our requirements. After that we perform convolution on the image and apply activation to the matrix. After every layer (number of layers may vary according to our needs), we apply batch normalization. Followed by pooling to reduce dimensionality size. We add as many convolutional layers until satisfied. After that the output is flattened and feed into a fully connected layer (FC Layer). The output class uses an activation function to classify images. We used the technique of back propagation till we reached our desired result. In back propagation, we used the concept of Adam optimization. Adam is an optimization algorithm that can be used to update network weights iterative based in training data.

Then to learn the concept of GAN. A GAN is composed of two smaller networks called the generator and discriminator. As the name suggests, the generator's task is to produce results that are indistinguishable from real data. The discriminator's task is to classify whether a sample came from the generator's model distribution or the original data distribution. Both of these subnetworks are trained simultaneously until the generator is able to consistently produce results that the discriminator cannot classify.

## 7.4 Peer Assessment Matrix

Here 1 represents the minimum rating and 5 represents the maximum rating of contribution of each member. The peer assessment matrix is shown in Table 6.

Table 6: Peer Assessment matrix

	Nimish Mathur	Shubham Goenka	Jadi Akhilesh Prasad	Dev Kevlani
Nimish Mathur	4	5	5	5
Shubham Goenka	5	5	4	5
Jadi Akhilesh Prasad	5	5	5	5
Dev Kevlani	5	5	5	4

## 7.5 Role Playing And Work Schedule

Table 7: Individual roles of group members

<b>Dev Kevlani</b>	Comparing GAN model with other models. Dataset selection and pre-processing of images.
<b>Shubham Goenka</b>	Dataset selection and pre-processing of images. Comparing GAN model with other models
<b>Jadi Akhilesh Prasad</b>	Deep Learning Implementation. Structuring model architecture.
<b>Nimish Mathur</b>	Structuring model architecture. Deep Learning Implementation.

Table 8: Work Schedule

[illegible]



## 7.6 Student Outcomes And Description and Performance

### Indicators (A-K Mapping)

The A-K mapping is described in table 9.

Table 9: A-K Mapping

SO	Description	Outcome
A1	Applying mathematical concepts to obtain analytical and numerical solutions.	<p>The training of GAN is done as a fight between generator and discriminator. This can be represented mathematically as</p> $\min_G \max_D V(D, G)$ $V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$

B1	Identify the constraints, assumptions and models for the problems.	<ul style="list-style-type: none"> <li>• User gives the input image according to the dataset on which the model is trained on.</li> <li>• Behavior of deep learning models outperforms any other models or methodology.</li> <li>• GPU computations using cudNN are optimized &amp; there is no need for further manual optimizations.</li> <li>• With random noise addition output confidence may change. Hence, an assumption of handling images having noise is avoided instead focus on simple cases is done.</li> <li>• GANs are highly unstable while working with high resolution images.</li> <li>• Implementation of GANs extending it into DCGANs.</li> </ul>
B2	Use appropriate methods, tools and techniques for data collection.	<ul style="list-style-type: none"> <li>• Convolutional Neural Networks</li> <li>• Artificial Neural Networks</li> <li>• Batch Normalization</li> <li>• Dropout</li> <li>• Max Pooling</li> <li>• Adam Optimization/Gradient Descent</li> </ul>

B3	Analyze and interpret results with respect to assumptions, constraints and theory.	<ul style="list-style-type: none"> <li>• The images from GAN had a clear visual improvement than those generated by the baseline CNN</li> <li>• One drawback was that the GAN tends to colorize objects in colors that are most frequently seen.</li> </ul>
C2	Can understand scope and constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.	The team has been able to gauge the social, economic and ethical benefits of this project such as persistence, learning from mistakes, development of originality and encouragement of a learning attitude.
D1	Fulfill assigned responsibility in multidisciplinary teams.	The whole project was divided among four team members to make effective use of the shared knowledge.
D2	Can play different roles as a team player.	Each member was easily able to switch from one responsibility to another responsibility to the need. Documentation and development responsibilities were shared among all the members.
E1	Identify engineering problems.	NIL
E2	Develop appropriate models to formulate solutions.	A DCGAN architecture model was developed to help proceed with the solution.
E3	Use analytical and computational methods to obtain solutions.	Tuning of hyper-parameters to obtain optimal architecture for the model.

F1	Showcase professional responsibility while interacting with peers and professional communities.	All of the members were punctual in attending the group meetings to discuss the upcoming responsibilities. The team was regular at arriving at the evaluation destination for panel and mentor evaluation as well.
G1	Produce a variety of documents such as laboratory or project reports using appropriate formats.	The team has been successful in preparing and presenting the project documentation in appropriate format.
G2	Deliver well-organized and effective oral presentation.	The team has been able to effectively communicate the idea behind the project and its implementation details to the mentor and the evaluation panel.
I1	Able to explore and utilize resources to enhance self-learning.	The team relied on different research papers on GANs and websites like <i>medium</i> , <i>quora</i> , <i>youtube</i> to help with the implementation and theoretical concepts of the project. MOOCs were also consulted to help set up the basics.
I2	Recognize the importance of life-long learning.	The team is motivated enough to learn more about deep CNN concepts like dropout, max pooling, batch normalization. Adam optimization, gradient descent, etc.
J1	Comprehend the importance of contemporary issues.	The team focused on the flaws of traditional evaluation and assignment methods which involved a lot of plagiarism and foul play. The idea was to help develop a better platform for the aforementioned purposes.
K2	Apply different data structures and algorithmic techniques.	Deep learning frameworks follow a traditional concept of forming a graph using pre-created modules. Hence, the whole architecture was formed taking into account the working of graphs.
K3	Use software tools necessary for computer engineering domain	Tensorflow, Keras, PyTorch frameworks were used in Python 2.0 while using CUDA with an Nvidia GPU on windows and linux machines.

## 7.7 Brief Analytical Assessment

Automatic synthesis of colored images from greyscale images would be interesting and useful, but current AI systems are still far from this goal. However, in recent years generic and powerful recurrent neural network architectures have been developed to learn discriminative text feature representations. Meanwhile, deep convolutional generative adversarial networks[3] have begun to generate highly compelling images of specific categories, such as faces, album covers, and room interiors. The automatic colorization of grayscale images has been an active area of research in machine learning for an extensive period of time. This is due to the large variety of applications such color restoration and image colorization for animations.

We developed a simple and effective GAN architecture and training strategy that enables compelling colored image synthesis from black and white images. We used the CIFAR-10 dataset with corresponding image descriptions using Generative Adversarial Network. The model is trained on this dataset, and we demonstrate its performance both on the training set categories and on the testing set. Hence, we demonstrated that the model can synthesize almost realistic colored images from the given input grayscale image.

### **Sources of information:**

The team started reading a number of research papers and blogs to explore the current problems which could be turned into the Capstone project. Talking to people around us as well gave us a number of ideas. Some of the ideas were discarded when their actual implementations were beyond our expertise or when they didn't fulfil the requirements of the capstone project. After many deliberations, Text to image generation using GANs was chosen as a capstone.

**Analytical, computation and/or experimental methods used:**

Proper working of our project and correct image generation for particular text was validated using validate dataset and refinement is done by tuning hyper-parameters.

Knowledge of engineering principles:

The successful completion of the project required basic knowledge of Deep learning concepts, Natural Language processing, Image Processing. In addition we also learnt about Pytorch, TensorFlow and Keras for our project.

**Sharing of responsibilities:**

To coordinate design and manufacturing dependencies, we divided the responsibilities among the four of us and used to meet at least once a week to update each other about the progress. Most of the work of the project was done on meeting. The documentation was shared through Google Docs among all the members for consistency.

**Appreciation for problem solving:**

Working on the capstone project gave us a simulation of the actual problem solving practices. We came to learn about the importance of proper documentation and requirement analysis. We also came to understand the importance of dividing up the task into different independent modules and linking them together at the end. This project also gave us deep insight of our subjects.

## APPENDIX A: REFERENCES

---

- [1] Goodfellow Ian J., Pouget-Abadie Jean , Mirza Mehdi , Xu Bing , Warde-Farley David, Ozair Sherjil, Courville Aaron, Bengio Yoshua . “Generative Adversarial Networks”. *Statistics Machine learning*, vol. 21(4), pp. 12-14, Tue, 10 Jun 2014 18:58:17 GMT.
- [2] Welsh T., Ashikhmin M., and Mueller K.. “Transferring color to greyscale images.” *ACM TOG*, volume 21, pp. 277–280, Jan 2002
- [3] Long J., Shelhamer E., and Darrell T.. “Fully convolutional networks for semantic segmentation.” *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, Dec 2015.
- [4] Levin A., Lischinski D., and Weiss Y.. “Colorization using optimization.” *ACM transactions on graphics (tog)*, volume 23, pp. 689–694, 2004.
- [5] Isola P., Zhu J., Zhou T., and Alexei A.. “Image-to-image translation with conditional adversarial networks.” *In IEEE Computer*, pp. 126-127, Mar 2016.
- [6] Geoffrey E. and Ruslan R.. „Reducing the dimensionality of data with neural networks”. *In TOG of science*, 313(5786): pp. 504–507, 2006.
- [7] Ronneberger O., Philipp F., and Thomas B.. “U-net: Convolutional networks for biomedical image segmentation.” *In International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer*, pp. 81-86, Jan 2015.
- [8] Salimans T., Goodfellow I, Zaremba W., Cheung V., Radford A., and Chen X.. “Improved techniques for training gans.” *In Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.
- [9] Andrew L., Awni Y., and Andrew N.. “Rectifier nonlinearities improve neural network acoustic models”. *Proc. ICML*, volume 30, pp. 11-16, Nov 2013.
- [10] Malinowski M. and Fritz M.. “A multi-world approach to question answering about realworld scenes based on uncertain input”. *Advances in Neural Information Processing Systems*, pp. 1682–1690, 2014.
- [11] Diederik K. and Jimmy B.. “Adam: A method for stochastic optimization.” *In ACM winter conference*, pp. 443-447, 2011.
- [12] Springenberg J., Dosovitskiy A., Thomas B., and Riedmiller M.. “Striving for simplicity: The all convolutional net”. *In ACM conference*, pp. 635-649, 2014.
- [13] Creswell A., White T., Dumoulin V, Arulkumaran K., Sengupta B., and Bharath A..

- “Generative adversarial networks: An overview.” *In IEEE International spring conference*, pp. 67-71, 2017.
- [14] Szegedy C., Vanhoucke V., Sergey I., Shlens J., and Zbigniew W.. “Rethinking the inception architecture for computer vision.” *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 34-38, 2016.
  - [15] Alon E., Feige U. and Feldman M.. “Max-Min Greedy Matching”. *Computer Science and Game Theory*. vol. 19(3), pp. 3-4, *Wed, 14 Mar 2018*
  - [16] Kaiming H., Xiangyu Z., Shaoqing R., and Jian S.. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.” *In Proceedings of the IEEE international conference on computer vision*, pp. 87-92, 2015
  - [17] Srivastava N., Hinton G., Krizhevsky A., Sutskever L., and Salakhutdinov R.. “Dropout: a simple way to prevent neural networks from overfitting.” *Journal of machine learning research*, 15(1), pp. 1929–1958, 2014
  - [18] Malinowski M. and Fritz M. “Towards a visual turing challenge.” *Artificial Intelligence*. vol. 3(1), pp. 2-8, 2015.
  - [19] Radford A., Metz L., and Soumith C.. “Unsupervised representation learning with deep convolutional generative adversarial networks.” *In ACM induction conference*, pp. 1189-1197, 2015
  - [20] Goodfellow I. and Malinowski M.. “Generative adversarial networks.” *In Nips 2016 tutorial*, pp 98-105, 2016.
  - [21] Sergey I. and Christian S.. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” *In International Conference on Machine Learning*, pp. 128-140, 2015.



## APPENDIX B: PLAGIARISM REPORT

### ORIGINALITY REPORT

18%

SIMILARITY INDEX

13%

INTERNET SOURCES

12%

PUBLICATIONS

%

STUDENT PAPERS

### PRIMARY SOURCES

1

[persagen.com](http://persagen.com)

Internet Source

4%

2

Han Zhang, Tao Xu, Hongsheng Li. "StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks", 2017 IEEE International Conference on Computer Vision (ICCV), 2017

Publication

2%

3

"Computer Vision – ECCV 2016", Springer Science and Business Media LLC, 2016

Publication

2%

4

[export.arxiv.org](http://export.arxiv.org)

Internet Source

1%

5

Hanqiao Huang, Yufei Zha, Meiyun Zheng, Peng Zhang. "ACFT: adversarial correlation filter for robust tracking", IET Image Processing, 2019

Publication

1%

6

[docplayer.net](http://docplayer.net)

Internet Source

1%

7	<a href="http://dspace.thapar.edu:8080">dspace.thapar.edu:8080</a> Internet Source	1 %
8	<a href="http://groups.csail.mit.edu">groups.csail.mit.edu</a> Internet Source	1 %
9	<a href="http://en.wikipedia.org">en.wikipedia.org</a> Internet Source	1 %
10	<a href="http://videlectures.net">videlectures.net</a> Internet Source	<1 %
11	<a href="http://arxiv.org">arxiv.org</a> Internet Source	<1 %
12	<a href="http://cdcu.ui.ac.id">cdcu.ui.ac.id</a> Internet Source	<1 %
13	Elyas Rashno, Ahmad Akbari, Babak Nasersharif. "A Convolutional Neural Network model based on Neutrosophy for Noisy Speech Recognition", 2019 4th International Conference on Pattern Recognition and Image Analysis (IPRIA), 2019 Publication	<1 %
14	<a href="http://dr.ntu.edu.sg">dr.ntu.edu.sg</a> Internet Source	<1 %
15	David J. Attokaren, Ian G. Fernandes, A. Sriram, Y. V. Srinivasa Murthy, Shashidhar G. Koolagudi. "Food classification from images using convolutional neural networks", TENCON	<1 %