Image Colorization and Enhancement using Generative Adversarial Network

Capstone Project Report

Mid Semester Evaluation

Submitted by:

(101803323) Nimish Mathur

(101803223) Shubham Goenka

(101803188) Jadi Akhilesh Prasad

(101803294) Dev Kevlani

COE

CPG No: 62

Under the Mentorship of

Dr. Surjit Singh

(Assistant Professor)

Dr. Singara Singh

(Associate Professor)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department
Thapar Institute of Engineering and Technology, Patiala
Month 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 GAN 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 6th semester (2020).

Date:

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

Counter Signed By:

Faculty Mentor: Dr. Surjit Singh
Assistant Professor
CSED,
TIET, Patiala

Co-Mentor: Dr. Singara Singh
Associate Professor
CSED,
TIET, Patiala

ACKNOWLEDGEMENT

We would like to express our thanks to our mentor(s) Dr Surjit Singh and Dr. Singara Singh.
He has been of great help in our venture, and an indispensable resource of technical knowledge. He is truly an amazing mentor 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:

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

TABLE OF CONTENTS

LIST OF TABLES

List of Figures

List of Abbreviations

CNN	Convolutional Neural Network
GAN	Generative Adversarial Neural Network
RNN	Recurrent Neural Network

Introduction

1.1Project Overview

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 (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 . 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.

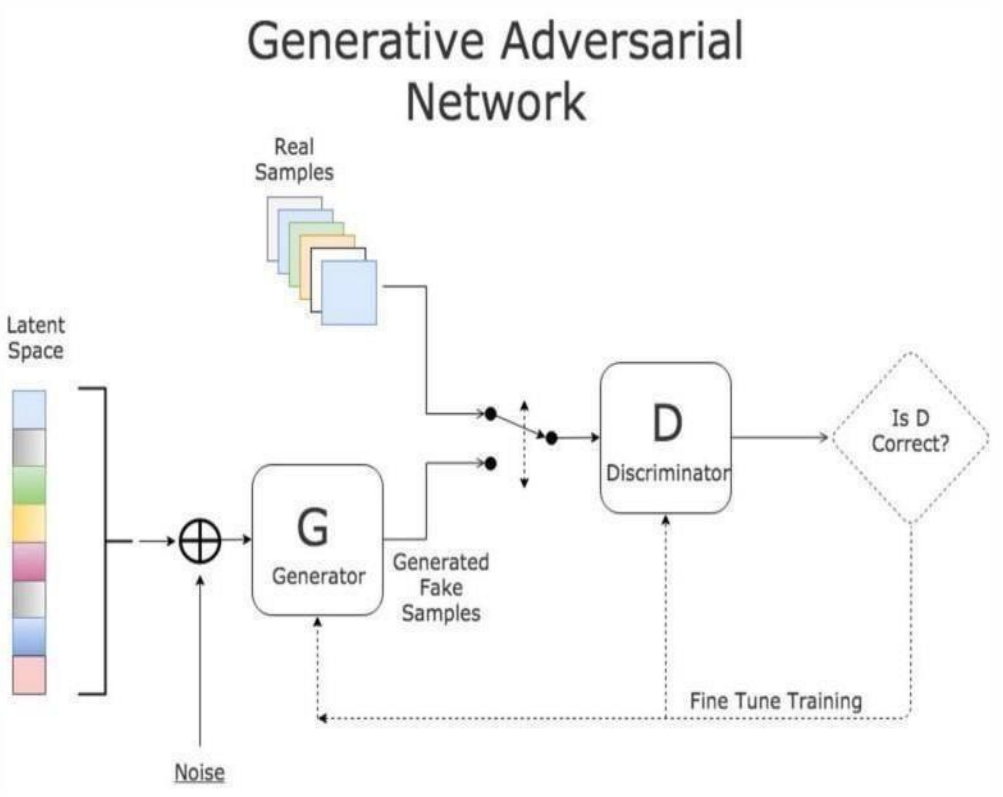


Figure 1: Generator and discriminator networks

1.1.1 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 Conditional Generative Adversarial Network (DCGAN).

1.1.2 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 a pixel-wise regression problem where structure in the input is highly aligned with structure in the output. That means the goal is to construct a network that not only generates an output with the same spatial dimension as the input, but also provides color information to each pixel in the grayscale input image.

1.1.3 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

Grayscale to colored image is usually visualized as a person's interesting or clear image which did 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 Problem

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 conditional deep contestive generative adversarial network (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.
- 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.

1.7 Approved Objectives

- Training a conditional generative adversarial network to generate realistic images from text descriptions. The result will show that generated images are consistent with input text descriptions.
- 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.
- Satellite images produced using GANS storage on cloud

1.8 Methodology

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 are 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.

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.

1.9 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.
- Generation of relatable image as per given rough sketch.

1.10 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 will also test our model by training it on the dataset of cars, animals and airplanes achieving sufficient accuracy.

2.0 Requirement Analysis

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

Generative adversarial networks (GANs) consist of a generator G and a discriminator D that compete in a two player minimax game : 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))).

Generative adversarial networks (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 . 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). 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:

$$\min_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \min_{\theta_G} -\mathbb{E}_z \left[\log(D(G(\mathbf{0}_z|x)))\right] + \lambda \|G(\mathbf{0}_z|x) - y\|_1$$

$$\max_{\theta_D} J^{(D)}(\theta_D, \theta_G) = \max_{\theta_D} \left(\mathbb{E}_y \left[\log(D(y|x))\right] + \mathbb{E}_z \left[\log(1 - D(G(\mathbf{0}_z|x)|x))\right]\right)$$

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.

2.1.1 Existing Systems & Solutions

Image Colorization Techniques: Models for the colorization of grayscales began back in the early 2000s. In 2002, Welsh et al. 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. 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, and trained on a reasonable amount of pictures of landscapes, achieves satisfactory results on this specific subset of pictures. It show that data augmentation significantly improves the performance and robustness of the model, and provide visual analysis of the prediction confidence.

.

2.1.2 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 consisten-t 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 Software Requirements Specification

2.2.1 Introduction

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.

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.

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.

2.2.2 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.

2.2.2.2 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) 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 and provide noise only in the form of dropout , 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 baselines contractive path: a series of 4 × 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.

2.2.3 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.

2.2.3.1 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.

2.2.3.2 Hardware Interfaces

Not Applicable.

2.2.3.3 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.

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.

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.

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.

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.

S.No.	Investigative Projects Techniques	Investigative Projects Description
1	Descriptive	● Research at conceptual level on GANs, DCGANs, Conditional GANs. ● Creating an optimal architecture of CNN to generate GANs.
2	Comparative	● Comparing different extensions of GANs to generate the best possible output for text to image conversion. ● Comparing different possible architectures and tuning the hyper-parameters.
3	Experimental	● Using machine learning/deep learning concepts on different frameworks, i.e. Keras, PyTorch.

Table 3: Investigative Techniques

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.

1.2 Work Breakdown Structure of image colorization using GANs

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

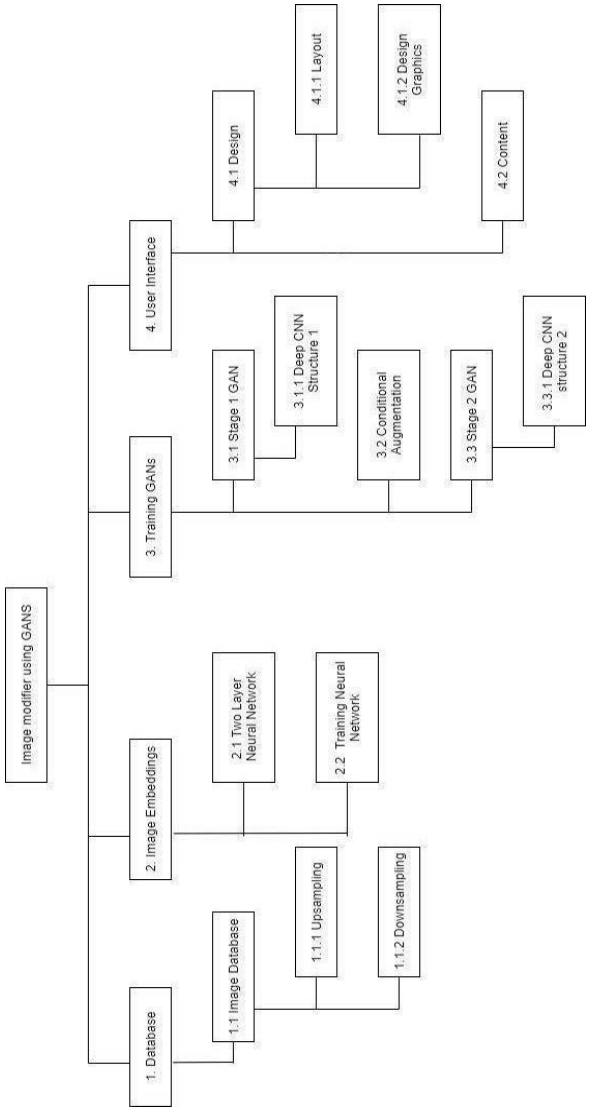


Figure 2: Work Breakdown Structure of Image Colorization using GANs

1.3 Tools & Technologies Used Technologies

- Artificial Neural Networks
- Convolutional Neural Networks
- Generative Adversarial Networks
- Parallel Training [12]
- Batch Normalization
- Dropout

Tools

- TensorFlow
- PyTorch
- CUDA
- Google Collab
- Numpy
- Scipy

DESIGN SPECIFICATIONS

4.1 System Architecture

4.1.1 MVC Architecture

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

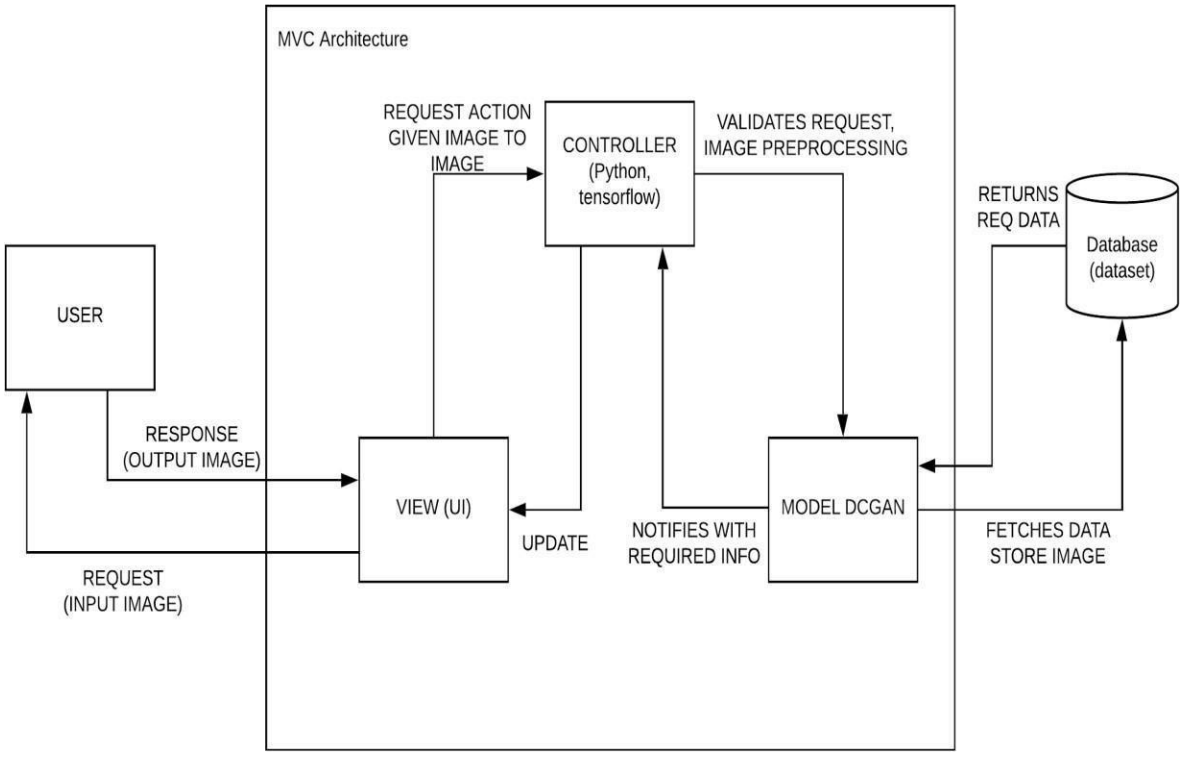
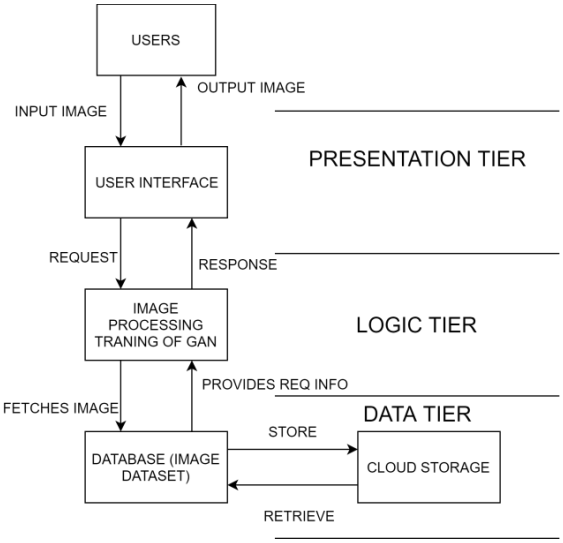


Figure 3: MVC Architecture

4.1.2 3 Tier Diagram



4.1.3 User Interface Diagram

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

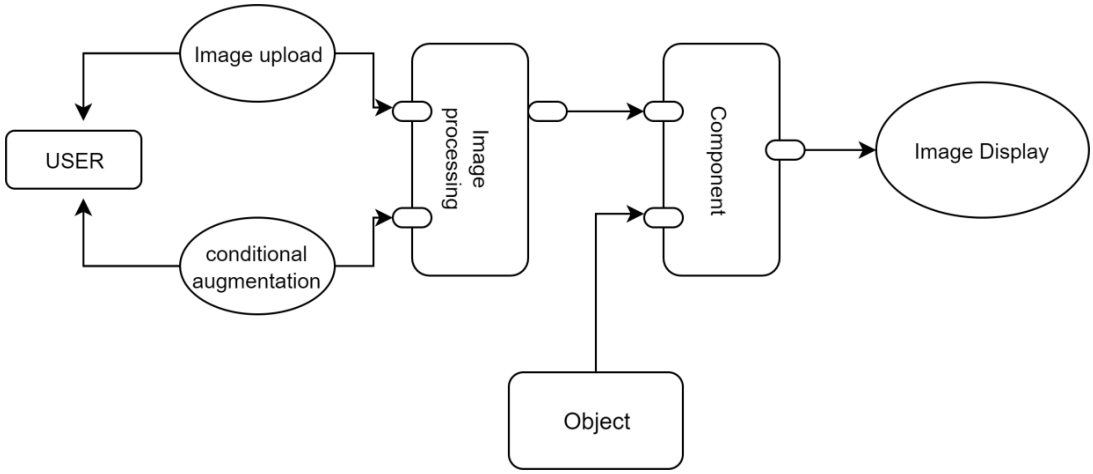


Figure 5: User Interface Diagram

4.2 Design Level Diagrams

4.2.1 Flowchart

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

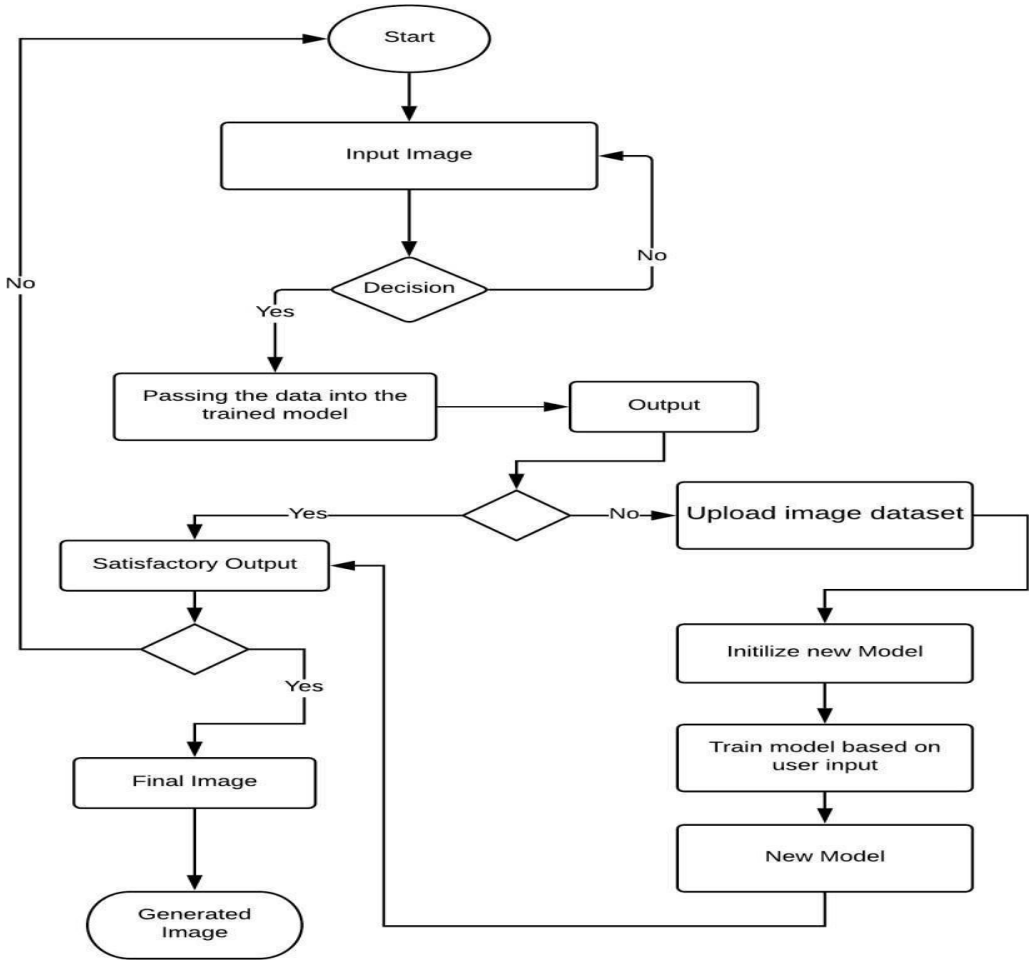


Figure 6: Activity Diagram

4.2.2 Sequence Diagram

The Sequence Diagram for the proposed project is shown below.

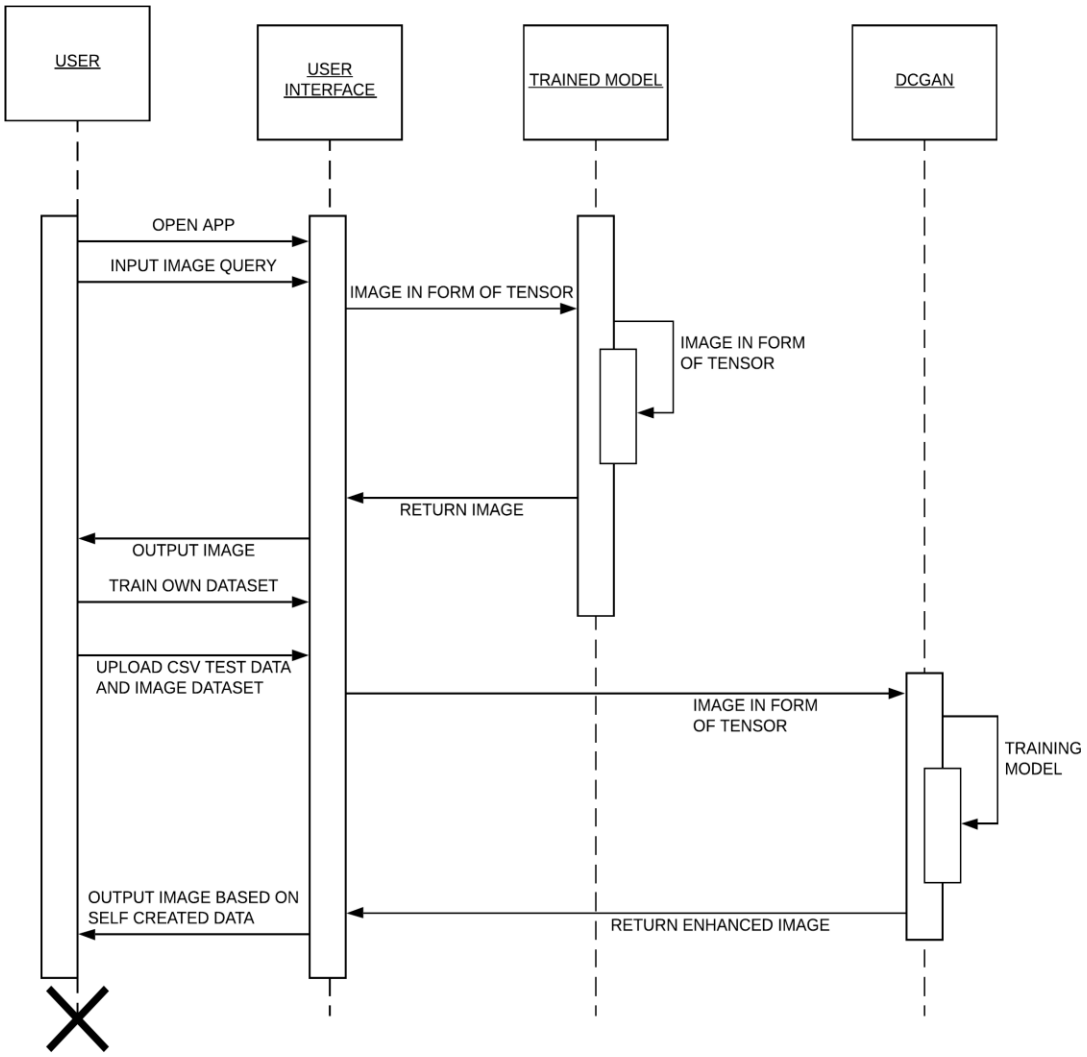


Figure 7: Sequence Diagram

4.2.3 Entity-Relationship Diagram

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

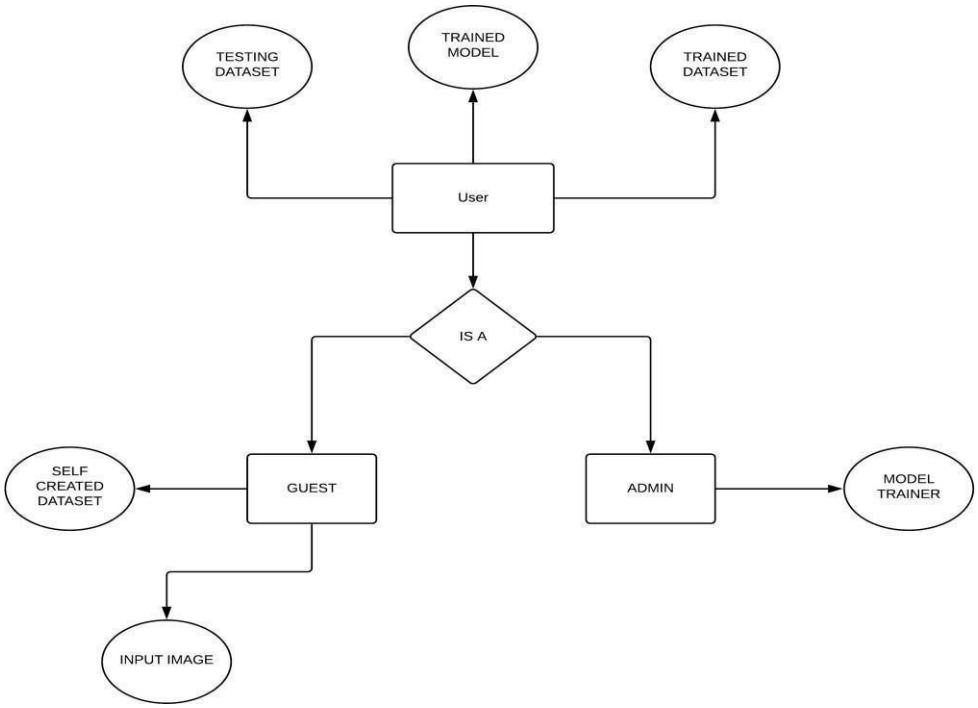


Figure 8: ER Diagram

4.3 User Interface Diagrams

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

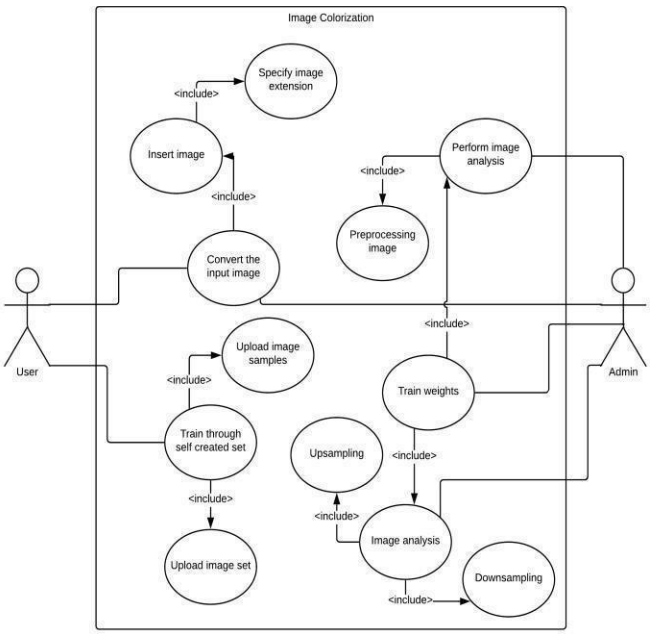


Figure 9:Use Case Diagram

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.

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[17].
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[18].
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:	Upsampling
Description:	The image is upsampled by using a Convolutional Neural Network appended to the GAN structure[15].
Actor:	Admin
Precondition:	Image should be available
Postcondition:	Upsampled image with better resolution is formed.

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

Postcondition:	A downsampled 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.

IMPLIMENTATION OF GANs

GANdressMNIST.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

```
[ ] !wget http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz
    !wget http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz
    !wget http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz
    !wget http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz
```

```
[ ] !ls
```

```
[ ] import numpy as np
    import tensorflow as tf
    import matplotlib.pyplot as plt
```

```
[ ] !pip install tensorflow==1.14.0
    import tensorflow as tf
    print(tf.__version__)
```

```
[ ] !mkdir MNIST_Fashion
    !cp *.gz MNIST_Fashion/
    from tensorflow.examples.tutorials.mnist import input_data
    mnist = input_data.read_data_sets("MNIST_Fashion/")
```

```
[ ] #Training PArams
    learning_rate = 0.0002
    batch_size = 128
    epochs = 100000

    #Network params
    image_dim = 784 #img sz is 28x28
    gen_hidd_dim = 256
    disc_hidd_dim  = 256
    z_noise_dim = 100

    def xavier_init(shape):
      return tf.random_normal(shape = shape, stddev= 1./tf.sqrt(shape[0]/2.0))
```

GANdressMNIST.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

```
[ ] weights = {
        "disc_H" : tf.Variable(xavier_init([image_dim, disc_hidd_dim])),
        "disc_final": tf.Variable(xavier_init([disc_hidd_dim,1])),
        "gen_H": tf.Variable(xavier_init([z_noise_dim, gen_hidd_dim])),
        "gen_final": tf.Variable(xavier_init([gen_hidd_dim, image_dim]))
    }

    bias = {
        "disc_H" : tf.Variable(xavier_init([disc_hidd_dim])),
        "disc_final": tf.Variable(xavier_init([1])),
        "gen_H": tf.Variable(xavier_init([gen_hidd_dim])),
        "gen_final": tf.Variable(xavier_init([image_dim]))
    }
```

```
[ ] def Discriminator(x):
      hidden_layer = tf.nn.relu(tf.add(tf.matmul(x, weights["disc_H"]), bias["disc_H"]))
      final_layer = (tf.add(tf.matmul(hidden_layer, weights["disc_final"]), bias["disc_final"]))
      disc_output = tf.nn.sigmoid(final_layer)
      return final_layer, disc_output
```

```
[ ] #Generator NW
    def Generator(x):
      hidden_layer = tf.nn.relu(tf.add(tf.matmul(x, weights["gen_H"]), bias["gen_H"]))
      final_layer = (tf.add(tf.matmul(hidden_layer, weights["gen_final"]), bias["gen_final"]))
      gen_output = tf.nn.sigmoid(final_layer)
      return gen_output
```

```
[ ] #define placeholders for external input

    z_input = tf.placeholder(tf.float32, shape = [None, z_noise_dim], name = "input_noise")
    x_input = tf.placeholder(tf.float32, shape = [None, image_dim], name = "real_noise")
```

GANdressMNIST.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

```
[ ] # building the GEN NW
    with tf.name_scope("Generator") as scope:
      output_Gen = Generator(z_input) #G(z)

     # Building the Disc NW
    with tf.name_scope("Generator") as scope:
      real_output1_Disc, real_output_disc = Discriminator(x_input) #implements D(x)
      fake_output1_Disc, fake_output_disc = Discriminator(output_Gen) # implements D(G(x))
```

```
[ ] #first kind of loss
    with tf.name_scope("Discriminator_Loss") as scope:
      Discriminator_Loss = -tf.reduce_mean(tf.log(real_output_disc+ 0.0001)+tf.log(1.- fake_output_disc+0.0001))

    with tf.name_scope("Genetator_Loss") as scope:
      Generator_Loss = -tf.reduce_mean(tf.log(fake_output_disc+ 0.0001)) # due to max log(D(G(x)))

      # T-board summary

      Disc_loss_total = tf.summary.scalar("Disc_Total_loss", Discriminator_Loss)
      Gen_loss_total = tf.summary.scalar("Gen_loss", Generator_Loss)
```

```
[ ] #second kind of loss
    with tf.name_scope("Discriminator_Loss") as scope:
      Disc_real_loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits = real_output1_Disc, labels = tf.ones_like(real_output1_Disc)))
      Disc_fake_loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits = fake_output1_Disc, labels = tf.zeros_like(fake_output1_Disc)))
      Discriminator_Loss = Disc_real_loss + Disc_fake_loss


    with tf.name_scope("Genetator_Loss") as scope:
      Generator_Loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits = fake_output1_Disc, labels = tf.ones_like(fake_output1_Disc)))

      # Tensorboarf summary
```

GANdressMNIST.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

```
[ ]   # Tensorboarf summary

    Disc_loss_real_summary = tf.summary.scalar('Disc_loss_real', Disc_real_loss)
    Disc_loss_fake_summary = tf.summary.scalar('Disc_loss_fake', Disc_fake_loss)
    Disc_loss_summary = tf.summary.scalar('Disc_total_loss', Discriminator_Loss)

    Disc_loss_total = tf.summary.merge([Disc_loss_real_summary, Disc_loss_fake_summary, Disc_loss_summary])
    Gen_loss_total = tf.summary.scalar('Gen_loss', Generator_Loss)
```

```
[ ] # Define the variables

    Generator_var = [weights["gen_H"], weights["gen_final"], bias["gen_H"], bias["gen_final"]]
    Discriminator_var = [weights["disc_H"], weights["disc_final"], bias["disc_H"], bias["disc_final"]]

    #Define the optimizer
    with tf.name_scope("Optimizer_Discriminator") as scope:
      Discriminator_optimize = tf.train.AdamOptimizer(learning_rate = learning_rate).minimize(Discriminator_Loss, var_list = Discriminator_var)

    with tf.name_scope("Optimizer_Generator") as scope:
      Generator_optimize = tf.train.AdamOptimizer(learning_rate = learning_rate).minimize(Generator_Loss, var_list = Generator_var)
```

```
# Initialize the variables

init = tf.global_variables_initializer()
sess = tf.Session()

sess.run(init)
writer = tf.summary.FileWriter("./log", sess.graph)

for epoch in range(epochs):
  x_batch, _ = mnist.train.next_batch(batch_size)

  #Generate noise to feed Discriminator
```

GANdressMNIST.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing

```
#Generate noise to feed Discriminator
z_noise = np.random.uniform(-1.,1.,size = [batch_size, z_noise_dim])
_, Disc_loss_epoch = sess.run([Discriminator_optimize, Discriminator_Loss], feed_dict = {x_input:x_batch, z_input:z_noise})
_, Gen_loss_epoch = sess.run([Generator_optimize, Generator_Loss], feed_dict = {z_input:z_noise})

#Running the Discriminator summary
summary_Disc_loss = sess.run(Disc_loss_total, feed_dict = {x_input:x_batch, z_input:z_noise})
# Adding the Discriminator summary
writer.add_summary(summary_Disc_loss, epoch)

#Running the Generator summary
summary_Gen_loss = sess.run(Gen_loss_total, feed_dict = {z_input:z_noise})
# Adding the Generator summary
writer.add_summary(summary_Gen_loss, epoch)

if epoch % 2000 == 0:
  print("Steps: {0}: Generator Loss: {1}, Discriminator Loss:{2}".format(epoch, Gen_loss_epoch, Disc_loss_epoch))
```

GANdressMNIST.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

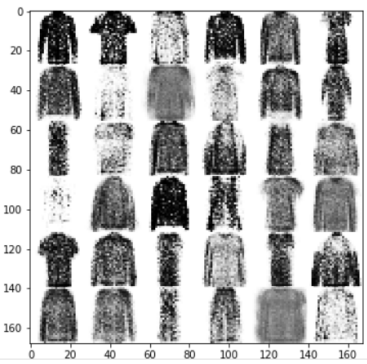
+ Code + Text

Connect Editing

```
[ ] #Testing
    # Generate images from noise, using the generator network

    n = 6
    canvas = np.empty((28*n, 28*n))
    for i in range(n):
      #Noise input
      z_noise = np.random.uniform(-1.,1., size = [batch_size, z_noise_dim])
      # Generate image from noise
      g = sess.run(output_Gen, feed_dict = {z_input:z_noise})
      # Reverse colors for better display
      g=-1*(g-1)
      for j in range(n):
        #Draw the generated digits
        canvas[i*28:(i+1)*28, j*28:(j+1)*28] = g[j].reshape([28, 28])

    plt.figure(figsize = (n,n))
    plt.imshow(canvas, origin = "upper", cmap = "gray")
    plt.show()
```



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 brown-ish 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.

REFERENCES

- https://www.researchgate.net/publication/325803914_Image_Colorization_Using_Generative_Adversarial_Networks
- https://www.arxiv-vanity.com/papers/1803.05400/
- https://heartbeat.fritz.ai/colorizing-b-w-images-with-gans-in-tensorflow-f444f737db6c
- http://cs231n.stanford.edu/reports/2017/pdfs/302.pdf
- https://link.springer.com/chapter/10.1007%2F978-981-16-0708-0_2
- https://towardsdatascience.com/colorizing-black-white-images-with-u-net-and-conditional-gan-a-tutorial-81b2df111cd8

END