GENERATING HUMAN FACES

A Project Report

Submitted in partial fulfilment of the
requirements of the award of the degree of

BACHELOR OF SCIENCE (COMPUTER SCIENCE)

By

HAIDER ZAINUDDIN ALI
Seat Number:

Under the esteemed guidance of
Prof. Javed Pathan Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE

RIZVI COLLEGE OF ARTS SCIENCE AND COMMERCE

Affiliated to University of Mumbai
MUMBAI, PIN 400050
MAHARASHTRA

2020-2021

RIZVI COLLEGE OF ARTS SCIENCE AND COMMERCE

(Affiliated to University of Mumbai)

MUMBAI-MAHARASHTRA-50

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the project entitled, "Protein cell localization", is benefited work of Haider Zainuddin Ali bearing Seat Number:__01___submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in COMPUTER SCIENCE from University of Mumbai.

Internal Guide

Co-ordinator

External Examiner

Date:

College Seal

ABSTRACT

The AI model will be trained on given dataset for this project I've used Human faces dataset [1] for Generating Human faces using GANs (Generative Adversarial Networks).

CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including

- 10,177 number of identities,
- 202,599 number of face images, and
- 5 landmark locations, 40 binary attributes annotations per image.

ACKNOWLEDGEMENT

I extremely grateful and remain indebted to our guide Prof. Javed Pathan for being a source of Inspiration and for his constant support in the Design, Implementation and Evolution of the Project. We are thankful to them for their constant constructive criticism and invaluable suggestions, which benefited us a lot while developing the project on “IDENTIFYING PROTEIN CELLS”.

I also thank my parents for supporting me and helping me in every way possible.

I also thank our Department teacher Prof. Arif Patel and who have helped in successful completion of the project.

I also offer our sincerest gratitude to our very own principal of college

Prof. Anjum Ara Ahmad for his constant support and consideration.

DECLARATION

I, hereby declare that the project entitled, “Identifying protein cells” done at RIZVI COLLEGE OF ARTS SCIENCE AND COMMERCE, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE (COMPUTER SCIENCE) to be submitted as fifth Semester project as part of our curriculum.

Haider Zainuddin Ali

_____

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

Today camera are so expensive to find patterns and to generate image in super resolution. That's where GAN's (Generative Adversarial Networks) comes in. It generates images which are almost close to be told as a real image. There are many applications of GAN's like Text-to-Image Translation, Face Frontal View Generation, Generate New Human Poses and many more. In this project I'll generate Human faces images given a Human faces dataset.

1.1 Background of the project

CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including 10,177 number of identities, 202,599 number of face images, and 5 landmark locations, 40 binary attributes annotations per image.

The dataset can be employed as the training and test sets for the following computer vision tasks: face attribute recognition, face detection, landmark (or facial part) localization, and face editing & synthesis.

1.2 Objective of the Project

Given a huge dataset of human faces generate a new set of human faces. To learn new patterns in human faces. Generative modelling involves using a model to generate new examples that plausibly come from an existing distribution of samples, such as generating new photographs that are similar but specifically different from a dataset of existing photographs.

A GAN is a generative model that is trained using two neural network models. One model is called the "*generator*" or "*generative network*" model that learns to generate new plausible samples. The other model is called the "*discriminator*" or "*discriminative network*" and learns to differentiate generated examples from real examples.

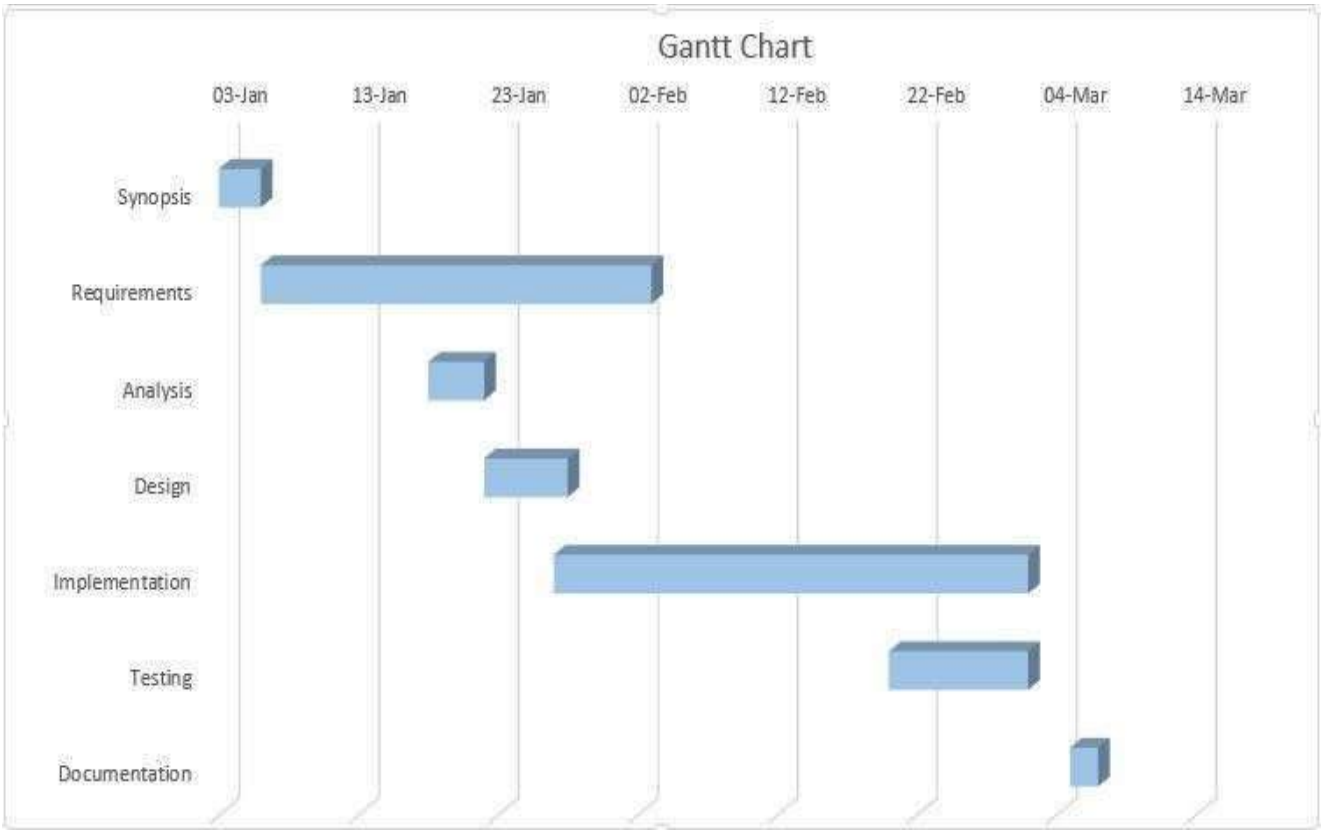
1.3 Purpose and Scope of the project

The generated images can be used to create a new dataset for learning new patterns or for classification task based on age, gender, etc.

1.4 Project Schedule

Gantt chart is type of BAR CHART, which depicts the chronological development of various phases in project. It sometime also refers as the schedule chart because of which a team can keep track of the project in terms of time scheduling.

A typical Gantt chart gives the start and finish dates of the various phases of development and displays it in the graphical form of bars.



CHAPTER 2. SYSTEM ANALYSIS

System analysis is defined as "the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way". Another view sees system analysis as a problem-solving technique that breaks down a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose.

The field of system analysis relates closely to requirements analysis or to operations research. It is also "an explicit formal inquiry carried out to help a decision maker identify a better course of action and make a better decision than she might otherwise have made."

System analysis is used in every field where something is developed. Analysis can also be a series of components that perform organic functions together, such as system engineering. System engineering is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed.

2.1 Proposed System

There is a Generator (Inverse CNN) and Discriminator (CNN) with predefined set of features with matrix know as latent features. The Generator keeps generating and learning new features from the given dataset and tries to defeat the Discriminator who purpose is to find how far is the generated image is from real image.

GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated). The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

GANs are an exciting and rapidly changing field, delivering on the promise of generative models in their ability to generate realistic examples across a range of problem domains, most notably in image-to-image translation tasks such as translating photos of summer to winter or day to night, and in generating photorealistic photos of objects, scenes, and people that even humans cannot tell are fake.

2.3. Hardware Requirements

Processor Type: Intel i3 or more, AMD E2-7110 APU
Processor Speed: 1.80 GHz or above

Hard disk Space: Minimum 1 GB free

RAM: 8 GB

Graphics: Nvidia Graphics (cuda Compatible)

2.4. Software Requirements

Operating System: Windows 7 or above, Linux.
Platform: Python IDLE v3.

Language: Python

2.5. Justification of Technology Used

Technologies used to develop this software are as follows:

An Overview of Windows10:

It is the successor to Windows8.1, and was released to manufacturing on July 15, 2015, and broadly released for retail sale on July 29, 2015. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

Windows 10 makes more consistent the user experience and functionality between different classes of device, and addresses shortcomings in the user interface that were introduced in Windows 8. Windows10 mobile, the successor to Windows Phone 8.1, shares some user interface elements and apps with its PC counterpart.The Windows Runtime app ecosystem was revised into the Universal Windows platform (UWP). These universal apps are made to run across multiple platforms and device classes, including smartphones, tablets, Xbox One consoles, and other compatible Windows 10 devices. Windows apps share code across platforms, have responsive designs that adapt to the needs of the device and available inputs, can synchronize data between Windows 10 devices (including notifications, credentials, and allowing cross-platform multiplayer for games), and are distributed through Microsoft store (rebranded from Windows Store since September 2017).

Python 3.7-32bit

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

IDLE has the following features:

- cross-platform: works mostly the same on Windows, Unix, and macOS.
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages.
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features.
- search within any window, replace within editor windows, and search through multiple files (grep).

debugger with persistent breakpoints, stepping, and viewing of global and local namespaces. configuration, browsers, and other dialogs.

CHAPTER 3. SYSTEM DESIGN

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. To find out the requirements for the new system, it is always essential to study and recognize the problems of existing system, which will become the building blocks of this development.

To study different alternatives and to get a viable solution, Study of the system is necessary.

Here are the stages performed in preliminary study of the system:

- Initially, when an enquiry comes, the data is saved in the database which can be later on follow up to convert it into a potential lead.
- When this potential student comes up, there is no need to fill in the data as it can be pulled up from the enquiry database.
- With the help of this software, everything (Data) in an organization can be digitally recorded into the system.
- Integrity can be maintained.

3.1 Methodology:

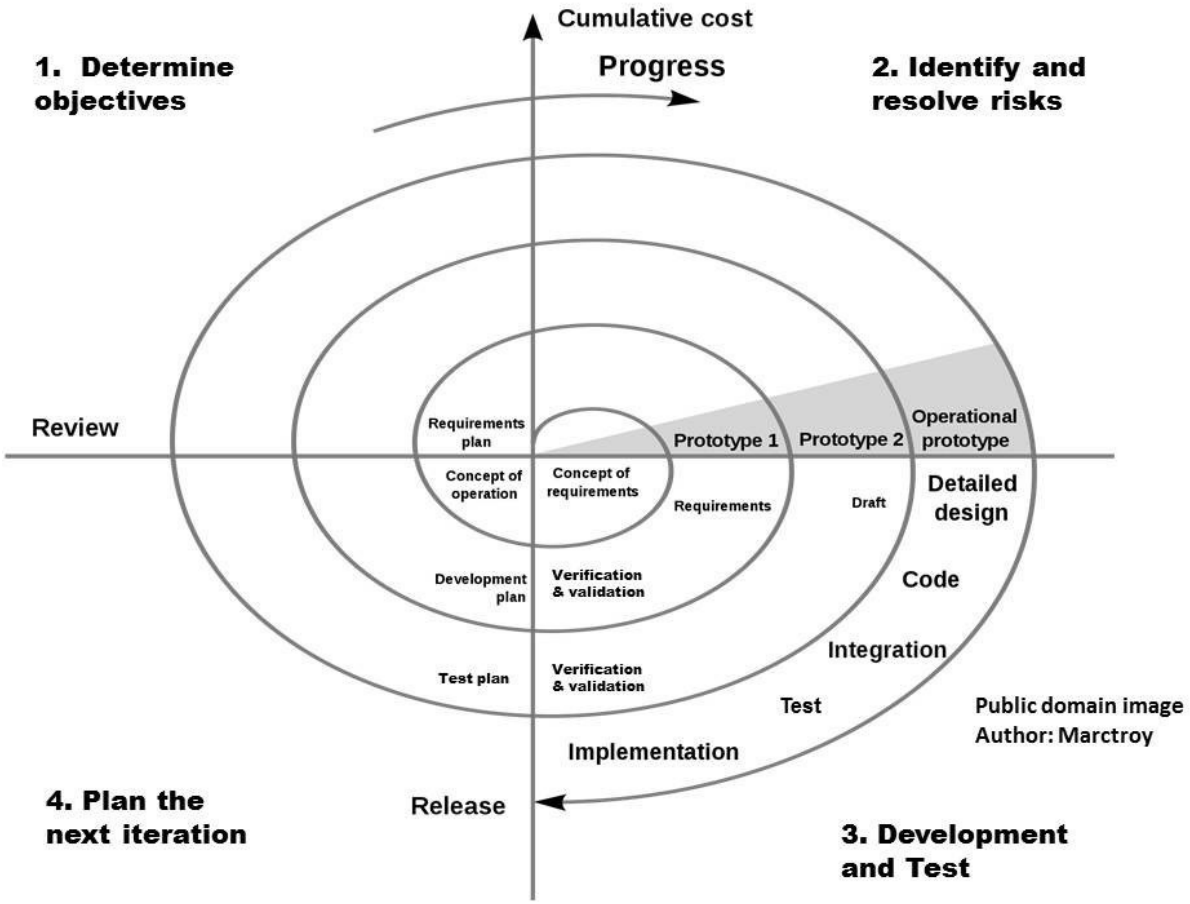
Software Development Model:

Spiral Model:

- A software development model serves the developer as a guiding tool during and throughout the development of the software, it facilitates to implement various steps / stages during the development of the overall project.
- Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project.
- Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project

risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using spiral model.

- The Radius of the spiral at any point represents the expenses (cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.



Spiral Model Approach:

Each phase of Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. Objectives determination and identify alternative solutions: Requirements are gathered from the customers and the objectives are identified, elaborated and analysed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
2. Identify and resolve Risks: During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.
3. Develop next version of the Product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
4. Review and plan for the next Phase: In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

3.2 Module Division

Here are the modules used in this project

Modules Used:

- Numpy:

  It is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices. Along with the high collection of mathematical functions to operate on these arrays.

- Pandas:

  Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

- Random:

  Python offers random module that can generate random numbers. These are pseudo-random number as the sequence of number generated depends on the seed.

- Matplotlib.pyplot:

  Matplotlib is a plotting library for the python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into
  applications using general-purpose GUI toolkits like TKinter, WxPython, Ot, orGTK+.

- Pytorch:

  PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab. It is free and open-source software released under the Modified BSD license.

- Torch.nn.DataParallel:

  Implements data parallelism at the module level.

This container parallelizes the application of the given `module` by splitting the input across the specified devices by chunking in the batch dimension (other objects will be copied once per device). In the forward pass, the module is replicated on each device, and each replica handles a portion of the input. During the backwards pass, gradients from each replica are summed into the original module.

The batch size should be larger than the number of GPUs used.

- Torch.optim:

It is a package implementing various optimization algorithms. Most commonly used methods are already supported, and the interface is general enough, so that more sophisticated ones can be also easily integrated in the future.

- Torch.nn:

It contains all the functions required to run deep neural networks.

- Torch.utils.data

At the heart of PyTorch data loading utility is the `torch.utils.data.DataLoader` class. It represents a Python iterable over a dataset, with support for

map-style and iterable-style datasets,
customizing data loading order,
single- and multi-process data loading,
automatic memory pinning.

- Torchvision.transforms

Transforms are common image transformations. They can be chained together using `Compose`. Additionally, there is the `torchvision.transforms.functional` module. Functional transforms give fine-grained control over the transformations. This is useful if you have to build a more complex transformation pipeline (e.g. in the case of segmentation tasks).

All transformations accept PIL Image, Tensor Image or batch of Tensor Images as input. Tensor Image is a tensor with `(C, H, W)` shape, where `C` is a number of channels, `H` and `W` are image height and width. Batch of Tensor Images is a tensor of `(B, C, H, W)` shape, where `B` is a number of images in the batch. Deterministic or random transformations applied on the batch of Tensor Images identically transform all the images of the batch.

- Torchvision.utils:

It contains utility function to modify images.

- Matplotlib.animation:

The easiest way to make a live animation in matplotlib is to use one of the Animation classes. In both cases it is critical to keep a reference to the instance object. The animation is advanced by a timer (typically from the host GUI framework) which the Animation object holds the only reference to. If you do not hold a reference to the Animation object, it (and hence the timers), will be garbage collected which will stop the animation.

To save an animation to disk use Animation.save or Animation.to_html5_video

- IPython.display import HTML:

Create a display object given raw data.

When this object is returned by an expression or passed to the display function, it will result in the data being displayed in the frontend. The MIME type of the data should match the subclasses used, so the Png subclass should be used for 'image/png' data. If the data is a URL, the data will first be downloaded and then displayed.

- Argparse:

The argparse module makes it easy to write user-friendly command-line interfaces. The program defines what arguments it requires, and argparse will figure out how to parse those out of sys.argv. The argparse module also automatically generates help and usage messages and issues errors when users give the program invalid arguments.

3.3 DATA

A popular component of computer vision and deep learning revolves around identifying faces for various applications from logging into your phone with your face or searching through surveillance images for a particular suspect. This dataset is great for training and testing models for face detection, particularly for recognising facial attributes such as finding people with brown hair, are smiling, or wearing glasses. Images cover large pose variations, background clutter, diverse people, supported by a large quantity of images and rich annotations.

202,599 number of face images of various celebrities
10,177 unique identities, but names of identities are not given
40 binary attribute annotations per image
5 landmark locations

Data Files:

img*align*celeba.zip: All the face images, cropped and aligned

list*eval*partition.csv: Recommended partitioning of images into training, validation, testing sets.

Images 1-162770 are training, 162771-182637 are validation, 182638-202599 are testing

list*bbox*celeba.csv: Bounding box information for each image. "x*1" and "y*1" represent the upper left point coordinate of bounding box. "width" and "height" represent the width and height of bounding box.

list*landmarks*align_celeba.csv: Image landmarks and their respective coordinates. There are 5 landmarks: left eye, right eye, nose, left mouth, right mouth

list*attr*celeba.csv: Attribute labels for each image. There are 40 attributes. "1" represents positive while "-1" represents negative

3.4 ER – Diagram

Entity Relationship Diagram:

E-R Model is a popular high level conceptual data model. This model and its variations are frequently used for the conceptual design of database application and many database design tools employ its concept. A database that confirms to an E-R diagram can be represented by a collection of tables in the relational system. The mapping of E-R diagram to the entities is:

Entity relationship diagram provides a visual starting point for database design that can also be used for determine information system requirements throughout an organization.

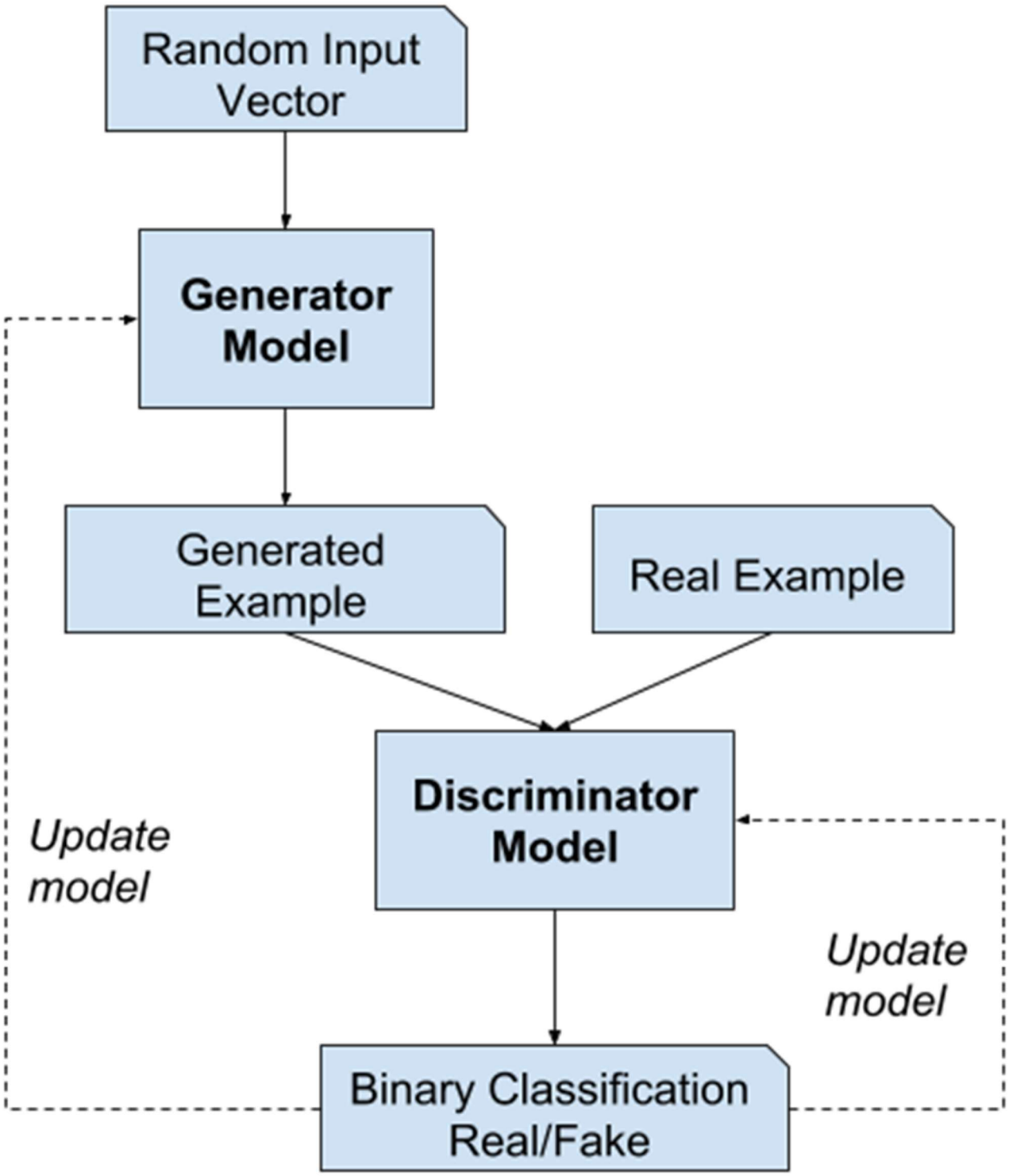
However, while an ERD can be helpful for organizing data that can be represented by a relational structure, it can't sufficiently represent semi-structured and unstructured data.

Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram. Terms that are simple and familiar always beats vague, technical-sounding words. In naming entities, remember to use singular nouns. However, adjectives may be used to distinguish entities belonging to the same class (part-time employee and full-time employee, for example). Meanwhile attribute names must be meaningful, unique, system-independent, and easily understandable.

Remove vague, redundant or unnecessary relationships between entities.

Never connect a relationship to another relationship.

Make effective use of colors. You can use colors to classify similar entities or to highlight key areas in your diagrams.

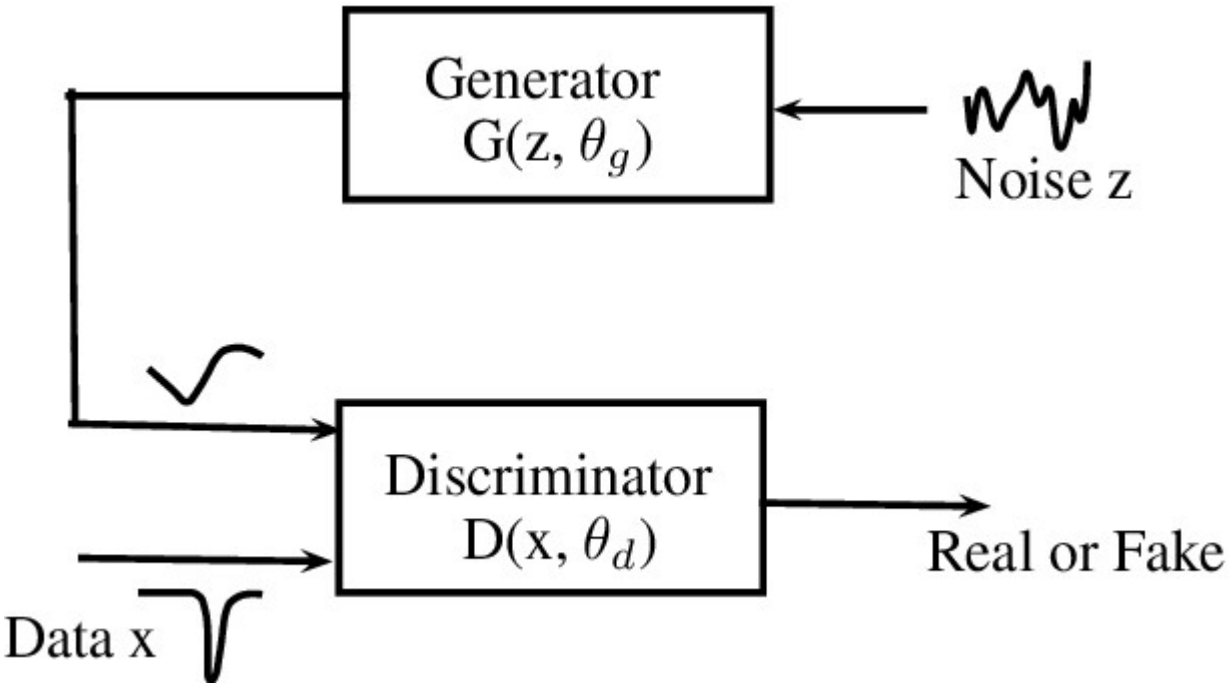

Random Input Vector
Generator Model
Generated Example
Real Example
Discriminator Model
Update model
Update model
Binary Classification Real/Fake

3.5 DFD

Dataflow Diagram:

Data flow diagram is the starting point of the design phase that functionally decomposes the requirements specification. A DFD consists of a series of bubbles joined by lines. The bubbles represent data transformation and the lines represent data flows in the system. A DFD describes what data flow rather than how they are processed, so it does not hardware, software and data structure. A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

A data flow diagram (DFD) is a significant modeling technique for analyzing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model. The data flow diagram is a graphical description of a system's data and how to Process transform the data is known as Data Flow Diagram (DFD). Unlike details flow chart, DFDs don't supply detail descriptions of modules that graphically describe a system's data and how the data interact with the system.



CHAPTER 4. IMPLEMENTATION AND CODING

Project implementation (or project execution) is the phase where visions and plans become reality. This is the logical conclusion, after evaluating, deciding, visioning, planning, applying for funds and finding the financial resources of a project.

Implementation simply means carrying out the activities described in your work plan. Executing a project in the water and sanitation sector is a very complex mission, as it requires the coordination of a wide range of activities, the overseeing of a team, the management of budget, the communication to the public, among other issues. Independent of whether it is a social project to raise the awareness and promote hygiene or it is a construction project for service delivery, there is a certain process that has to be followed.

The following description will give you an introduction into the implementation of projects in sustainable sanitation and water management, and highlights key aspects that have to be taken into account for a successful implementation.

It is important to take into account that independently of the nature of the project, implementation takes time, usually more than it is planned, and that many external constraints can appear, which should be considered when initiating the implementation step (i.e. seasonality in availability of community engagement/resources)

The experiment is as previously described aimed at evaluating the prediction performance of given algorithms for predicting movie ratings, and most importantly to investigate whether or not any significant advantages of using RF can be found.

The following sections will go through the process of collecting, preparing and mining the movie data in order to reliably evaluate the predictive performance of these algorithms. This includes selection of datasets as well as selection and preparation of data and attributes, but also the practical process of acquiring the results as in selection of metrics, configuration of parameters for the algorithms and design of the experiment itself.

1. Data Collection:

Data collection is very important and costly part of any project. In order to properly evaluate the relevant deep learning algorithms on protein labels predictions a number of relevant datasets has to be considered for the experiment.

The CelebA dataset is available for **non-commercial research purposes** only.

All images of the CelebA dataset are obtained from the Internet which are not property of MMLAB, The Chinese University of Hong Kong. The MMLAB is not responsible for the content nor the meaning of these images.

2. Data Preparation

The data is in a zip compressed folder and is separated into different directories. For this we need to make a dataset which will contain images with its unique image id to make it easy for the model to train and make better predictions. Pytorch Dataset and Dataloader class helps us to make custom datasets. Dataloader helps to load the whole dataset in cuda (gpu from NVIDIA). GPU helps model to train much faster than cpu. Dataset and Dataloader class helps to helps to randomly split the dataset into equal size of batches. Also they use more cpu blocks to distribute their work.

4.1 Code:

4.1.1 Importing Libraries:

```
from __future__ import print_function
#%matplotlib inline
import argparse
import os
import random
import torch
import torch.nn as nn
import torch.nn.parallel
import torch.backends.cudnn as cudnn
```

```
import torch.optim as optim
import torch.utils.data
import torchvision.datasets as dset
import torchvision.transforms as transforms
import torchvision.utils as vutils
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from IPython.display import HTML
```

4.1.2 Defining Input Variables

```
# Root directory for dataset
dataroot = "/content/celeba-dataset/img_align_celeba"

# Number of workers for dataloader
workers = 2

# Batch size during training
batch_size = 128

# Spatial size of training images. All images will be resized to this
#   size using a transformer.
image_size = 64

# Number of channels in the training images. For color images this is 3
nc = 3

# Size of z latent vector (i.e. size of generator input)
nz = 100

# Size of feature maps in generator
ngf = 64

# Size of feature maps in discriminator
ndf = 64

# Number of training epochs
num_epochs = 10

# Learning rate for optimizers
lr = 0.0002

# Beta1 hyperparam for Adam optimizers
beta1 = 0.5

# Number of GPUs available. Use 0 for CPU mode.
ngpu = 1
```

4.1.3 Creating Custom Pytorch Dataset and loading it into cuda (if available)

```
dataset = dset.ImageFolder(root=dataroot,
                           transform=transforms.Compose([
                               transforms.Resize(image_size),
                               transforms.CenterCrop(image_size),
                               transforms.ToTensor(),
                               transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
                           ]))
# Create the dataloader
dataloader = torch.utils.data.DataLoader(dataset, batch_size=batch_size,
                                         shuffle=True, num_workers=workers)

# Decide which device we want to run on
device = torch.device("cuda:0" if (torch.cuda.is_available() and ngpu > 0) else "cpu")
```

4.1.4 Visualizing batch of images

```
real_batch = next(iter(dataloader))
plt.figure(figsize=(8,8))
plt.axis("off")
plt.title("Training Images")
plt.imshow(np.transpose(vutils.make_grid(real_batch[0].to(device)[:64], padding=2,
normalize=True).cpu(),(1,2,0)))
```

4.1.5 Initializing Weights

```
# custom weights initialization called on netG and netD
def weights_init(m):
    classname = m.__class__.__name__
    if classname.find('Conv') != -1:
        nn.init.normal_(m.weight.data, 0.0, 0.02)
    elif classname.find('BatchNorm') != -1:
        nn.init.normal_(m.weight.data, 1.0, 0.02)
        nn.init.constant_(m.bias.data, 0)
```

4.1.6 Defining Discriminator and loading it to cuda (Nividia GPU)

```
class Discriminator(nn.Module):
    def __init__(self, ngpu):
        super(Discriminator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is (nc) x 64 x 64
            nn.Conv2d(nc, ndf, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf) x 32 x 32
            nn.Conv2d(ndf, ndf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 2),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*2) x 16 x 16
            nn.Conv2d(ndf * 2, ndf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 4),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*4) x 8 x 8
            nn.Conv2d(ndf * 4, ndf * 8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 8),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*8) x 4 x 4
            nn.Conv2d(ndf * 8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )

    def forward(self, input):
        return self.main(input)



# Create the Discriminator
netD = Discriminator(ngpu).to(device)

# Handle multi-gpu if desired
if (device.type == 'cuda') and (ngpu > 1):
    netD = nn.DataParallel(netD, list(range(ngpu)))

# Apply the weights_init function to randomly initialize all weights
#  to mean=0, stdev=0.2.
netD.apply(weights_init)

# Print the model
print(netD)
```

4.1.7 Defining Generator and loading it to cuda (Nividia GPU)

```
# Generator Code

class Generator(nn.Module):
    def __init__(self, ngpu):
        super(Generator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is Z, going into a convolution
            nn.ConvTranspose2d( nz, ngf * 8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(ngf * 8),
            nn.ReLU(True),
            # state size. (ngf*8) x 4 x 4
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(True),
            # state size. (ngf*4) x 8 x 8
            nn.ConvTranspose2d( ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(True),
            # state size. (ngf*2) x 16 x 16
            nn.ConvTranspose2d( ngf * 2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(True),
            # state size. (ngf) x 32 x 32
            nn.ConvTranspose2d( ngf, nc, 4, 2, 1, bias=False),
            nn.Tanh()
            # state size. (nc) x 64 x 64
        )

    def forward(self, input):
        return self.main(input)

# Create the generator
netG = Generator(ngpu).to(device)

# Handle multi-gpu if desired
if (device.type == 'cuda') and (ngpu > 1):
    netG = nn.DataParallel(netG, list(range(ngpu)))

# Apply the weights_init function to randomly initialize all weights
#  to mean=0, stdev=0.2.
netG.apply(weights_init)

# Print the model
print(netG)
```

4.1.8 Defining Loss and optimizer

```
# Initialize BCELoss function
criterion = nn.BCELoss()

# Create batch of latent vectors that we will use to visualize
#  the progression of the generator
fixed_noise = torch.randn(64, nz, 1, 1, device=device)

# Establish convention for real and fake labels during training
real_label = 1.
fake_label = 0.

# Setup Adam optimizers for both G and D
optimizerD = optim.Adam(netD.parameters(), lr=lr, betas=(beta1, 0.999))
optimizerG = optim.Adam(netG.parameters(), lr=lr, betas=(beta1, 0.999))
```

4.1.9 Training Step

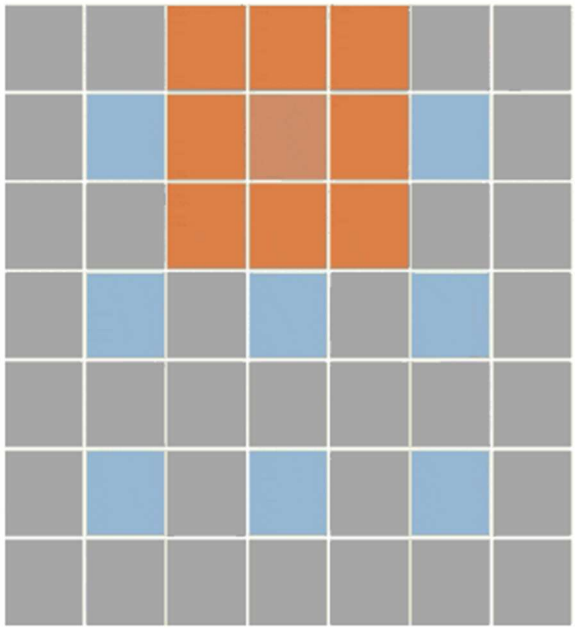
```
# Training Loop

# Lists to keep track of progress
img_list = []
G_losses = []
D_losses = []
iters = 0

print("Starting Training Loop...")
# For each epoch
for epoch in range(num_epochs):
    # For each batch in the dataloader
    for i, data in enumerate(dataloader, 0):

        ############################
        # (1) Update D network: maximize log(D(x)) + log(1 - D(G(z)))
        ###########################
        ## Train with all-real batch
        netD.zero_grad()
        # Format batch
        real_cpu = data[0].to(device)
        b_size = real_cpu.size(0)
        label = torch.full((b_size,), real_label, dtype=torch.float, device=device)
        # Forward pass real batch through D
        output = netD(real_cpu).view(-1)
        # Calculate loss on all-real batch
        errD_real = criterion(output, label)
        # Calculate gradients for D in backward pass
```

```
errD_real.backward()
D_x = output.mean().item()

## Train with all-fake batch
# Generate batch of latent vectors
noise = torch.randn(b_size, nz, 1, 1, device=device)
# Generate fake image batch with G
fake = netG(noise)
label.fill_(fake_label)
# Classify all fake batch with D
output = netD(fake.detach()).view(-1)
# Calculate D's loss on the all-fake batch
errD_fake = criterion(output, label)
# Calculate the gradients for this batch
errD_fake.backward()
D_G_z1 = output.mean().item()
# Add the gradients from the all-real and all-fake batches
errD = errD_real + errD_fake
# Update D
optimizerD.step()

############################
# (2) Update G network: maximize log(D(G(z)))
###########################
netG.zero_grad()
label.fill_(real_label)  # fake labels are real for generator cost
# Since we just updated D, perform another forward pass of all-fake batch through D
output = netD(fake).view(-1)
# Calculate G's loss based on this output
errG = criterion(output, label)
# Calculate gradients for G
errG.backward()
D_G_z2 = output.mean().item()
# Update G
optimizerG.step()

# Output training stats
if i % 50 == 0:
    print('[%d/%d][%d/%d]\tLoss_D: %.4f\tLoss_G: %.4f\tD(x): %.4f\tD(G(z)): %.4f / %.4f'
          % (epoch, num_epochs, i, len(dataloader),
             errD.item(), errG.item(), D_x, D_G_z1, D_G_z2))

# Save Losses for plotting later
G_losses.append(errG.item())
D_losses.append(errD.item())

# Check how the generator is doing by saving G's output on fixed_noise
if (iters % 500 == 0) or ((epoch == num_epochs-1) and (i == len(dataloader)-1)):
    with torch.no_grad():
```

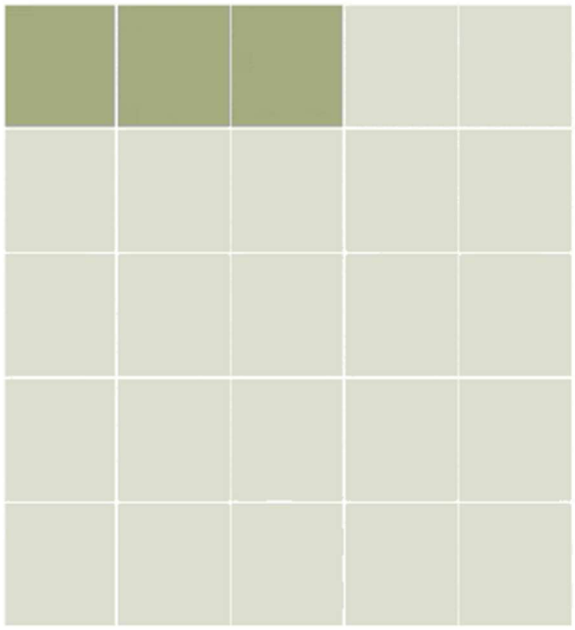
```
        fake = netG(fixed_noise).detach().cpu()
    img_list.append(vutils.make_grid(fake, padding=2, normalize=True))

iters += 1
```

4.2 Screenshots of the project

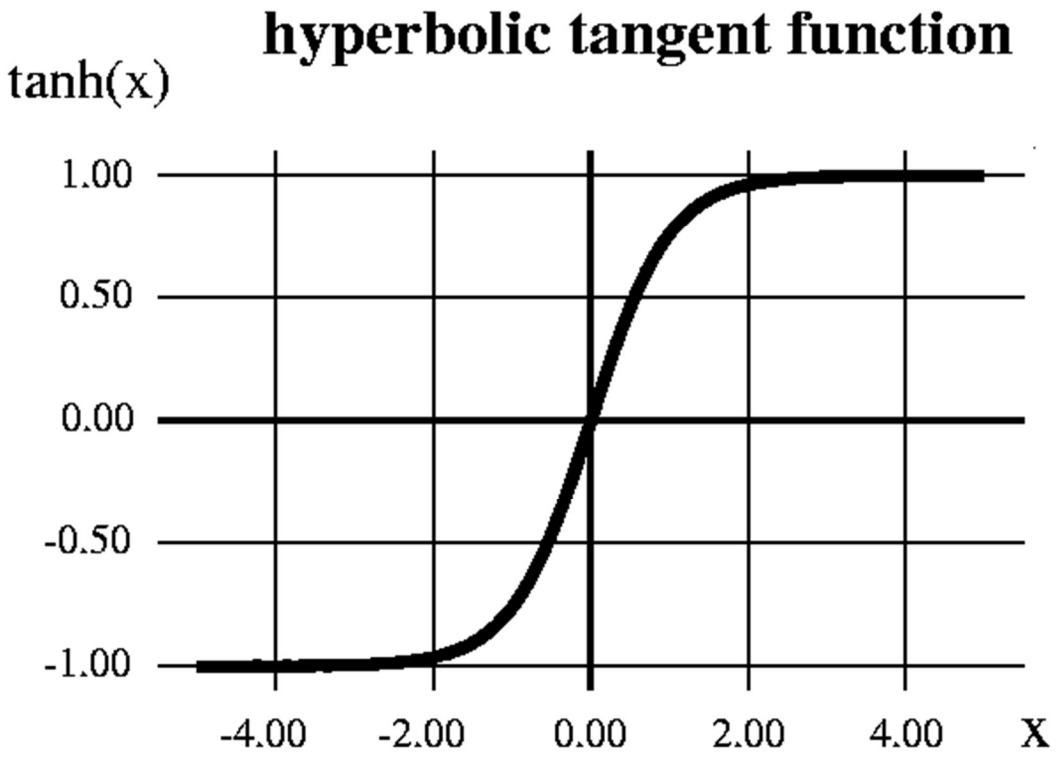
Generator



Input



Output

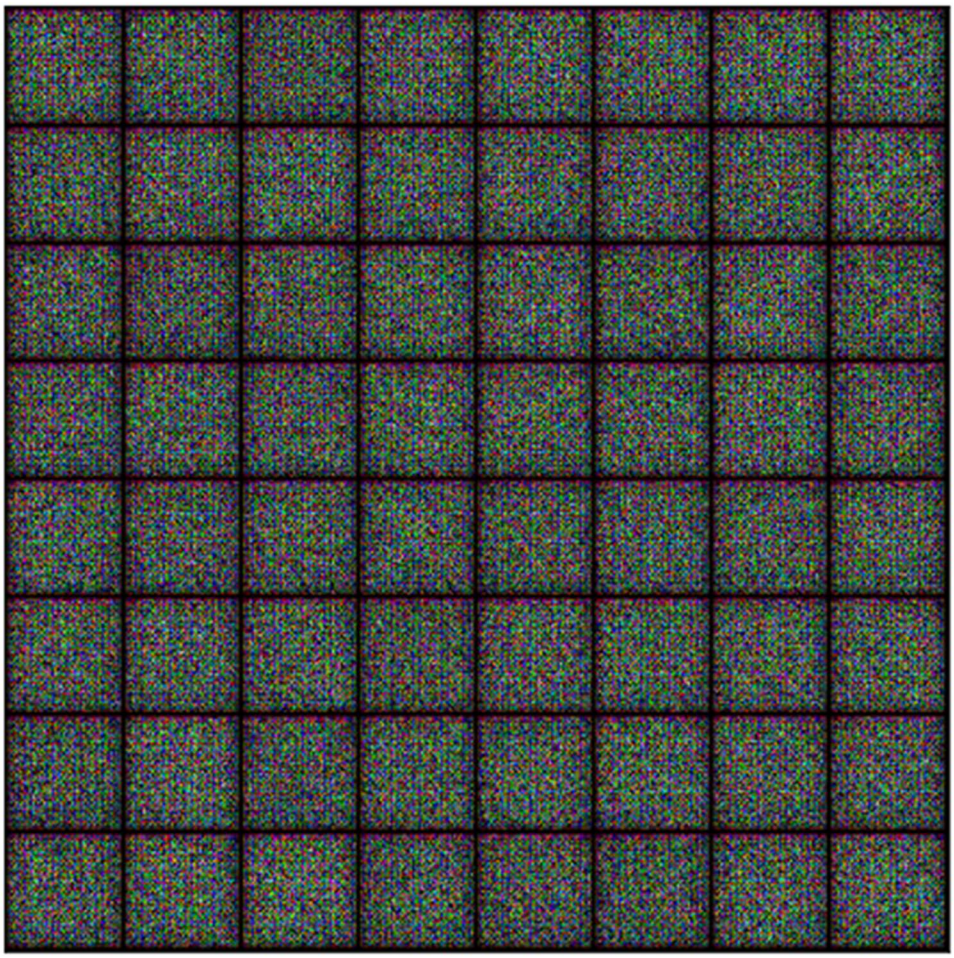


Binary Cross Entropy

$$-\frac{1}{N}\sum_{i=1}^{N} y_i log(h_\theta(x_i)) + (1 - y_i) log(1 - h_\theta(x_i))$$

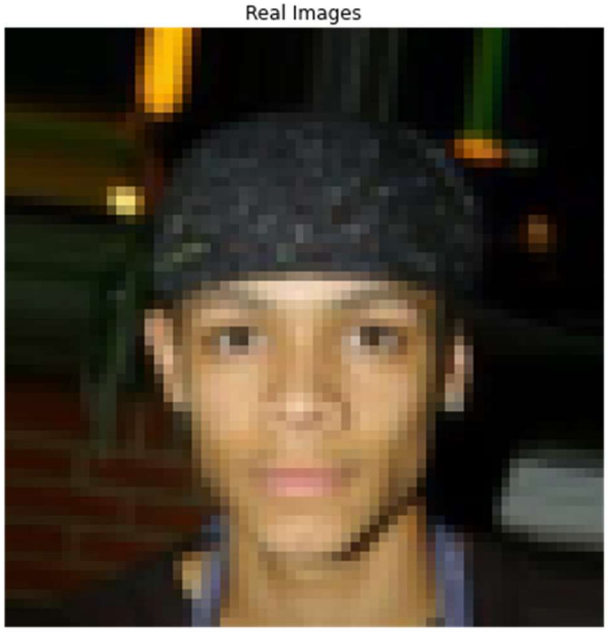
Intuition: make our model's distribution as close as possible to the (unknown) true distribution. In other words, the model which maximizes the probability of the training data is the best model.

Latent feature

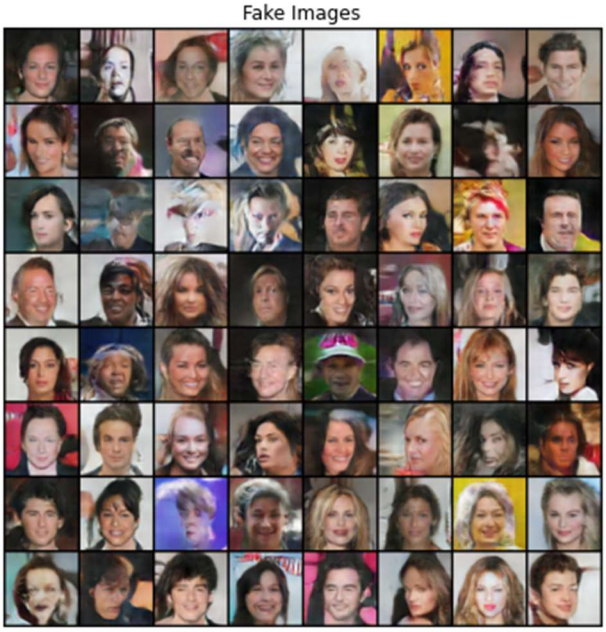


Real vs Fake Images

Real Images



Fake Images



4.3 Testing Approach

Testing is vital for the success of any software. No system design is ever perfect. Testing is also carried in two phases. First phase is during the software engineering that is during the module creation. Second phase is after the completion of software. This is system testing which verifies that the whole set of programs hanged together.

White Box Testing:

In this technique, the close examination of the logical parts through the software is tested by cases that exercise species sets of conditions or loops. All logical parts of the software checked once. Errors that can be corrected using this technique are typographical errors, logical expressions which should be executed once may be getting executed more than once and error resulting by using wrong controls and loops. When the box testing tests all the independent part within a module a logical decisions on their true and the false side are exercised, all loops and bounds within their operational bounds were exercised and internal data structure to ensure their validity were exercised once.

Black Box Testing:

This method enables the software engineer to device sets of input techniques that fully exercise all functional requirements for a program. Black box testing tests the input, the output and the external data.

It checks whether the input data is correct and whether we are getting the desired output.

Unit Testing:

Each module is considered independently. It focuses on each unit of software as implemented in the source code. It is white box testing.

Implementation and Software Specification Testing:

Detailed Design of Implementation

This phase of the systems development life cycle refines hardware and software specifications, establishes programming plans, trains users and implements extensive testing procedures, to evaluate design and operating specifications and/or provide the basis for further modification.

Technical Design

This activity builds upon specifications produced during new system design, adding detailed technical specifications and documentation.

Test Specifications and Planning

This activity prepares detailed test specifications for individual modules and programs, job streams, subsystems, and for the system as a whole.

Programming and Testing

This activity encompasses actual development, writing, and testing of program units or modules.

User Training

This activity encompasses writing user procedure manuals, preparation of user training materials, conducting training programs, and testing procedures.

Acceptance Test

A final procedural review to demonstrate a system and secure user approval before a system becomes operational.

Installation Phase

In this phase the new computerized system is installed, the conversion to new procedures is fully implemented, and the potential of the new system is explored.

System Installation

The process of starting the actual use of a system and training user personnel in its operation.

Review Phase

This phase evaluates the successes and failures during a systems development project, and to measure the results of a new Computerized Tran system in terms of benefits and savings projected at the start of the project.

Development Recap

A review of a project immediately after completion to find successes and potential problems in future work.

Post-Implementation Review

A review, conducted after a new system has been in operation for some time, to evaluate actual system performance against original expectations and projections for cost-benefit improvements. Also identifies maintenance projects to enhance or improve the system.

THE STEPS IN THE SOFTWARE TESTING

The steps involved during Unit testing are as follows:

a. Preparation of the test cases.

b. Preparation of the possible test data with all the validation checks.

c. Complete code review of the module.

d. Actual testing done manually.

e. Modifications done for the errors found during testing.

f. Prepared the test result scripts.

The unit testing done included the testing of the following items:

1. Functionality of the entire module/forms.

2. Validations for user input.

3.Checking of the Coding standards to be maintained during coding.

1. Testing the module with all the possible test data.

2. Testing of the functionality involving all type of calculations etc.
3. Commenting standard in the source files.

After completing the Unit testing of all the modules, the whole system is integrated with all its dependencies in that module. While System Integration, We integrated the modules one by one

and tested the system at each step. This helped in reduction of errors at the time of the system testing.

The steps involved during System testing are as follows:

1. Integration of all the modules/forms in the system.
2. Preparation of the test cases.
3. Preparation of the possible test data with all the validation checks.
4. Actual testing done manually.
5. Recording of all the reproduced errors.
6. Modifications done for the errors found during testing.
7. Prepared the test result scripts after rectification of the errors.

The System Testing done included the testing of the following items:

1. Functionality of the entire system as a whole.
2. User Interface of the system.
3. Testing the dependent modules together with all the possible test data scripts.
4. Verification and Validation testing.
5. Testing the reports with all its functionality.

After the completion of system testing, the next following phase was the Acceptance Testing. Clients at their end did this and accepted the system with appreciation. Thus, we reached the final phase of the project delivery. There are other six tests, which fall under special category. They are described below:

Peak Load Test:

It determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand. For example, test the system by activating all terminals at the same time.

Storage Testing:

It determines the capacity of the system to store transaction data on a disk or in other files.

Performance Time Testing: it determines the length of time system used by the system to process transaction data. This test is conducted prior to implementation to determine how long it takes to get a response to an inquiry, make a backup copy of a file, or send a transmission and get a response.

Recovery Testing: This testing determines the ability of user to recover data or re-start system after failure. For example, load backup copy of data and resume processing without data or integrity loss.

Procedure Testing:

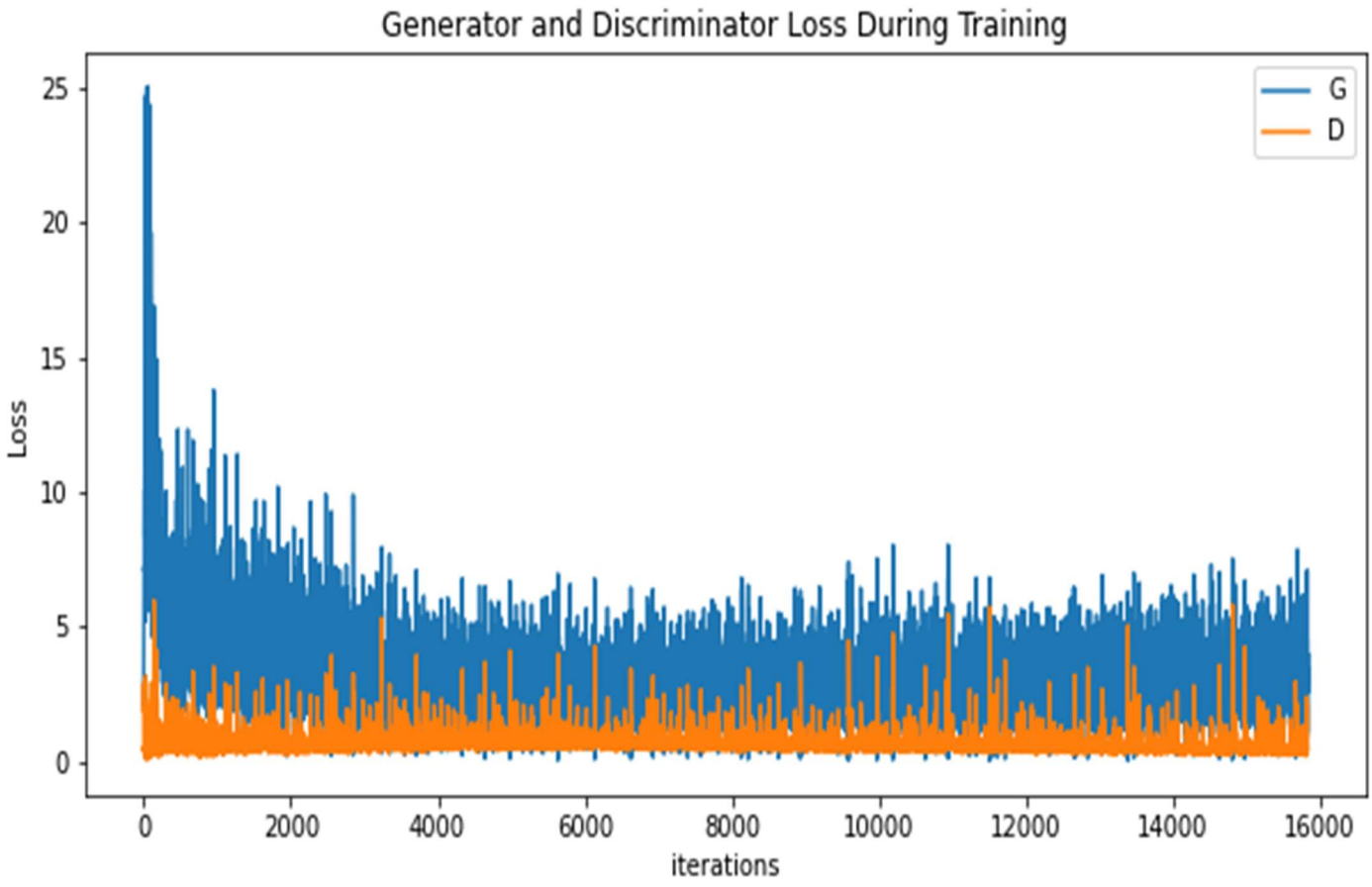
It determines the clarity of documentation on operation and uses of system by having users do exactly what manuals request. For example, powering down system at the end of week or responding to paper-out light on printer. Human Factors Testing: It determines how users will use the system when processing data or preparing reports

4.4 Test Cases

Test step	Test data	Expected Results	Actual Results	Pass/Fail
Data collection	Different images collected through internet.	Data gathered Succesfull	Data gathered Succesful	Pass
Data preparation	Images are fitted in zip file and csv is made to access it easily in Pytorch and make a custom dataset.	Reads successful	Reads Successful	Pass
Data Mining	Algorithms such as ANN, CNN, Generator, Discriminator.	Successfull y performed	Successfull y performed	Pass

CHAPTER 5. CONCLUSION AND FUTURE WORK

This data is collected by Chinese university of Honk Kong as a project. I have done hyperparameters to make Fake images look close to real.



5.1. Limitation of the system

The proposed system has some limitation Generator takes lot epochs to train and get the results close to Real images. My next step will be to study StyleGANand apply on this dataset.

CHAPTER 6. REFERENCES

[1] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[2] https://pytorch.org/docs/stable/index.html

[3] https://jovian.ai