CHAPTER 1:
INTRODUCTION

1.1.1 Background

The crucial job of a Data Scientist is to collect create data as much as possible so that we can train the model and get better accuracy. Used to solve real world problems, pre-collected data is not useful. Insufficient data may lead to low accuracy or inefficient use of the model. ML-based Uber and Google’s self-driving cars are trained with the use of synthetic data .In a research department, the use of synthetic data facilitates the development and delivery of innovative products, especially in cases where the required data may not be readily accessible.

Brief history of Technology/concept

In 1959, David Hubel and Torsten Wiesel described the "simple cells" and "complex cells" of the human visual cortex. They suggested that both types of cells be used for pattern recognition. A "simple cell" responds to edges and bars in a particular direction. "Complex cells" also respond to edges and bars in a particular direction, but differ from simple cells in that these edges and bars can be moved in the scene and the cells continue to respond. For example, a simple cell may only respond to the horizontal bar at the bottom of the image, and a complex cell may respond to the horizontal bar at the bottom, center, or top of the image. This property of complex cells is called "spatial invariance".

1.2 Applications

In a research department, the use of synthetic data facilitates the development and delivery of innovative products, especially in cases where the required data may not be readily accessible. The crucial job of a Data Scientist is to collect create data as much as possible so that we can train the model and get better accuracy. Used to solve real world problems, pre-collected data is not useful. Insufficient data may lead to low accuracy or inefficient use of the model.

1.3 Research motivation and Problem statement

Automatic synthesis of realistic images is extremely difficult task and even the state-of-the-art AI/ML algorithm suffer to fulfil this expectation. Privacy, Training, Testing, Using the Generated images for sales in stores

Problem statement

To construct an efficient generator that utilizes AI/ML algorithms to generate the desired outputs.

1.4 Primary objectives

To Generate a Photo Realistic Images using GAN(Generative Adversarial Networks) and use it to advertise products.

2.1 INTRODUCTION

GANs have emerged as a potent tool for generating images that find application in advertising campaigns. Leveraging their capacity to learn from extensive datasets and produce visually compelling and authentic content, GANs prove valuable in delivering realistic and high-quality visual assets, GANs offer a unique opportunity to create eye-catching and appealing visuals tailored specifically for advertising purposes. Using a GAN for generating images for ads involves training a generator network on a dataset of relevant images, such as product images or images associated with the desired advertising theme. The generator learns to create new images that resemble the training data, capturing the visual patterns and style present in the original images.

2.2 RELATED WORK

A lot of research has been done in various aspects of GAN. The first paper to introduce the GAN was [1]. Later on adaptations were made based on the paper[1] which proved a next stage of development. All these methods use new architecture manipulation. The papers [2] – [7] were huge leap forward in the generation of images..

CHAPTER 4:

DESIGN

4.1 Architectural Design

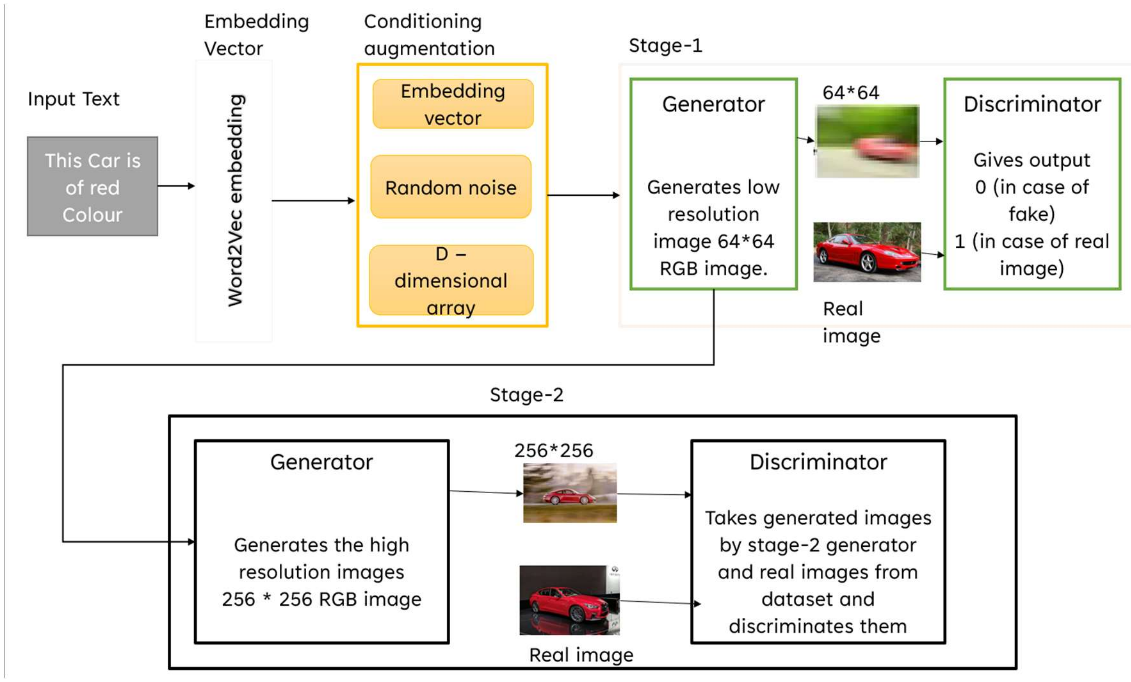

Fig 4.1 Architectural Design

4.2 General overview of proposed work

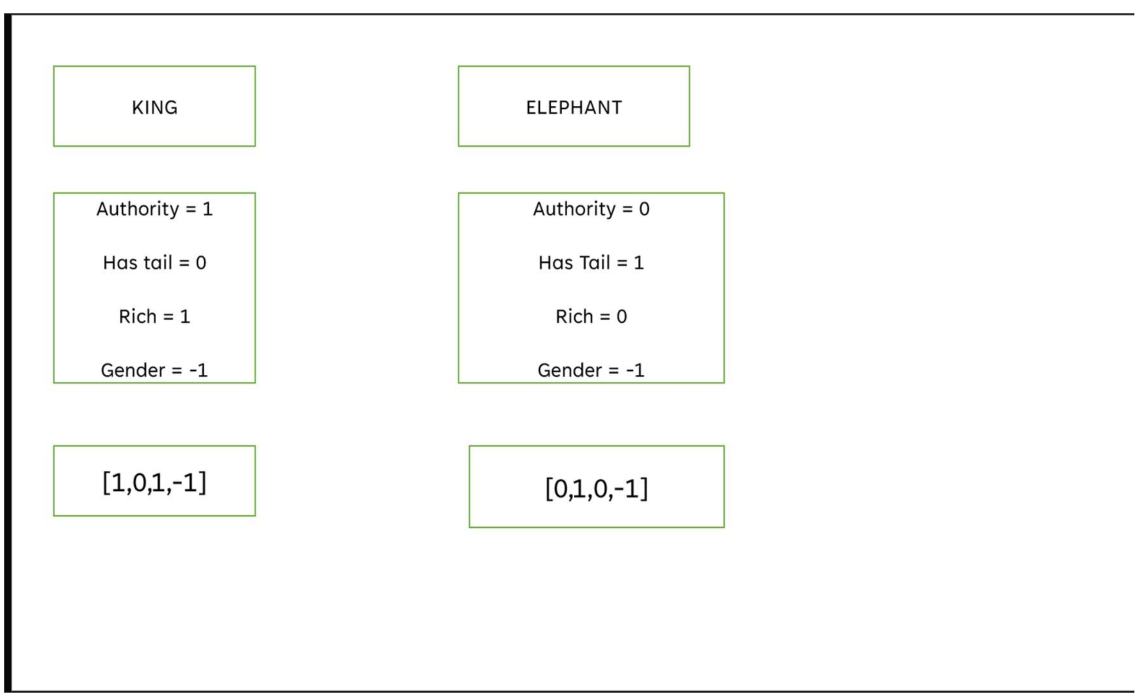

Fig 4.2.1 words to numbers representation

	Horse	King	Man	Queen	...	Woman
Authority	0	1	0.2	1	...	0.2
Has tail	1	0	0	0	...	0
Rich	0	1	0.3	1	...	0.2
Gender	-1	-1	-1	1	..	1

$$\underset{\text{King}}{\begin{bmatrix} 1 \\ 0 \\ 1 \\ -1 \end{bmatrix}} - \underset{\text{Man}}{\begin{bmatrix} 0.2 \\ 0 \\ 0.3 \\ -1 \end{bmatrix}} + \underset{\text{Woman}}{\begin{bmatrix} 0.2 \\ 0 \\ 0.2 \\ 1 \end{bmatrix}} = \begin{bmatrix} 1 \\ 0 \\ 0.9 \\ 1 \end{bmatrix} \sim \underset{\text{Queen}}{\begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}}$$

Fig 4.2.2 List of words and their features

CHAPTER 5:

IMPLEMENTATION

PSEUDOCODE:

- **Importing libraries and dependencies and reading dataset:**

```
import gensim
import re
import pandas as pd
```

READ FROM DATASET

```
df1 = pd.read_csv("/content/drive/MyDrive/Final year project/Datasets/final_data.csv")
df1.head()
```

	Unnamed: 0	Review
0	0	Wow... Loved this place.
1	1	Crust is not good.
2	2	Not tasty and the texture was just nasty.
3	3	Stopped by during the late May bank holiday of...
4	4	The selection on the menu was great and so wer...

Checking for any NaN values in very important because NaN values are representes as float so it cannot preprocess the data

Fig 5.1

- Preprocessing

Checking for any NaN values in very important because NaN values are representes as float so it cannot preprocess the data

```
df1.shape
df1['Review'].isnull().sum()
df = df1.dropna()
df['Review'].isnull().any()
```

```
False
```

PRE PROCESSING

```
review_text = df.Review.apply(gensim.utils.simple_preprocess)
review_text
```

```
0                                [wow, loved, this, place]
1                                  [crust, is, not, good]
2         [not, tasty, and, the, texture, was, just, nasty]
3         [stopped, by, during, the, late, may, bank, ho...
4         [the, selection, on, the, menu, was, great, an...
                                ...
228601    [works, great, just, like, my, original, one, ...
228602    [great, product, great, packaging, high, quali...
228603    [this, is, great, cable, just, as, good, as, t...
228604    [really, like, it, becasue, it, works, well, w...
228605    [product, as, described, have, wasted, lot, of...
Name: Review, Length: 227662, dtype: object
```

Fig 5.2

- Gensim Model

```
[ ] model = gensim.models.Word2Vec(window=10,min_count = 2,workers=4)

[ ] model.build_vocab(review_text,progress_per=1000)

model.corpus_count

227662

[ ] model.train(review_text,total_examples=model.corpus_count,epochs = 25)

(200412086, 272710635)

[ ] model.save("/content/drive/MyDrive/Final year project/word2vec.model")

[ ] loaded_model = gensim.models.Word2Vec.load("/content/drive/MyDrive/Final year project/word2vec.model")

[ ] model.wv.similarity(w1="cheap",w2="bad")

0.5289325

[ ] model.wv.most_similar("yellow")

[('orange', 0.8269873857498169),
 ('green', 0.816033124923706),
 ('grey', 0.813794732093811),
```

Fig 5.3

- Comparison

```
[ ] model.wv.similarity(w1="cheap",w2="bad")

0.5289325

model.wv.most_similar("yellow")

[('orange', 0.8269873857498169),
 ('green', 0.816033124923706),
 ('grey', 0.813794732093811),
 ('gray', 0.7980300784111023),
 ('red', 0.7960870265960693),
 ('brown', 0.7892323732376099),
 ('lavender', 0.7858045697212219),
 ('purple', 0.7848584055900574),
 ('yellowish', 0.7815498113632202),
 ('teal', 0.7810453176498413)]
```

Fig 5.4

- Implementation of gan model

```python
if __name__ == '__main__':
    data_dir = "/content/drive/MyDrive/Finalyearproject/Datasets/birds/birds"
    start = 100
    resume = True
    train_dir = data_dir + "/train"
    test_dir = data_dir + "/test"
    image_size = 64
    batch_size = 64
    z_dim = 100
    stage1_generator_lr = 0.002
    stage1_discriminator_lr = 0.002
    stage1_lr_decay_step = 600
    epochs = 201
    condition_dim = 128

    embeddings_file_path_train = train_dir + "/char-CNN-RNN-embeddings.pickle"
    embeddings_file_path_test = test_dir + "/char-CNN-RNN-embeddings.pickle"

    filenames_file_path_train = train_dir + "/filenames.pickle"
    filenames_file_path_test = test_dir + "/filenames.pickle"

    class_info_file_path_train = train_dir + "/class_info.pickle"
    class_info_file_path_test = test_dir + "/class_info.pickle"

    cub_dataset_dir = "/content/drive/MyDrive/Finalyearproject/Datasets/CUB_200_2011/CUB_200_2011"

    # Define optimizers
    dis_optimizer = Adam(lr=stage1_discriminator_lr, beta_1=0.5, beta_2=0.999)
    gen_optimizer = Adam(lr=stage1_generator_lr, beta_1=0.5, beta_2=0.999)

    """
```

Fig 5.5

- Gan model

```python
if resume==False:
  ca_model = build_ca_model()
  ca_model.compile(loss="binary_crossentropy", optimizer="adam")

  stage1_dis = build_stage1_discriminator()
  stage1_dis.compile(loss='binary_crossentropy', optimizer=dis_optimizer)

  stage1_gen = build_stage1_generator()
  stage1_gen.compile(loss="mse", optimizer=gen_optimizer)

  embedding_compressor_model = build_embedding_compressor_model()
  embedding_compressor_model.compile(loss="binary_crossentropy", optimizer="adam")

  adversarial_model = build_adversarial_model(gen_model=stage1_gen, dis_model=stage1_dis)
  adversarial_model.compile(loss=['binary_crossentropy', KL_loss], loss_weights=[1, 2.0],
                            optimizer=gen_optimizer, metrics=None)

else:
  ca_model = build_ca_model()
  ca_model.compile(loss="binary_crossentropy", optimizer="adam")

  stage1_dis = build_stage1_discriminator()
  stage1_dis.compile(loss='binary_crossentropy', optimizer=dis_optimizer)

  stage1_gen = build_stage1_generator()
  stage1_gen.compile(loss="mse", optimizer=gen_optimizer)

  embedding_compressor_model = build_embedding_compressor_model()
  embedding_compressor_model.compile(loss="binary_crossentropy", optimizer="adam")

  adversarial_model = build_adversarial_model(gen_model=stage1_gen, dis_model=stage1_dis)
  adversarial_model.compile(loss=['binary_crossentropy', KL_loss], loss_weights=[1, 2.0],
```

Fig 5.6

- Training

```python
for epoch in range(start,epochs):
    print("========================================")
    print("Epoch is:", epoch)
    print("Number of batches", int(X_train.shape[0] / batch_size))

    gen_losses = []
    dis_losses = []

    # Load data and train model
    number_of_batches = int(X_train.shape[0] / batch_size)
    for index in range(number_of_batches):
        print("Batch:{}".format(index+1))

        """
        Train the discriminator network
        """
        # Sample a batch of data
        z_noise = np.random.normal(0, 1, size=(batch_size, z_dim))
        image_batch = X_train[index * batch_size:(index + 1) * batch_size]
        embedding_batch = embeddings_train[index * batch_size:(index + 1) * batch_size]
        image_batch = (image_batch - 127.5) / 127.5 # makes the image darker and gets it in range -1 to 1

        # Generate fake images
        fake_images, _ = stage1_gen.predict([embedding_batch, z_noise], verbose=3)

        # Generate compressed embeddings
        compressed_embedding = embedding_compressor_model.predict_on_batch(embedding_batch) #converts the 1024 size to much more less i.e 128
        compressed_embedding = np.reshape(compressed_embedding, (-1, 1, 1, condition_dim)) #-1 is whatever size should be calculated by numpy itself
        compressed_embedding = np.tile(compressed_embedding, (1, 4, 4, 1))

        dis_loss_real = stage1_dis.train_on_batch([image_batch, compressed_embedding],
                                                  np.reshape(real_labels, (batch_size, 1)))
```

Fig 5.7

- Saving the model

```python
        g_loss = adversarial_model.train_on_batch([embedding_batch, z_noise, compressed_embedding],[K.ones((batch_size, 1)) * 0.9, K.ones((batch_size, 256)) * 0.9
        print("g_loss:{}".format(g_loss))

        dis_losses.append(d_loss)
        gen_losses.append(g_loss)

        """
        Save losses to Tensorboard after each epoch
        """
        # write_log(tensorboard, 'discriminator_loss', np.mean(dis_losses), epoch)
        # write_log(tensorboard, 'generator_loss', np.mean(gen_losses[0]), epoch)

        # Generate and save images after every 2nd epoch
        if epoch % 5 == 0 and index%20==0:
            # z_noise2 = np.random.uniform(-1, 1, size=(batch_size, z_dim))
            z_noise2 = np.random.normal(0, 1, size=(batch_size, z_dim))
            embedding_batch = embeddings_test[0:batch_size]
            fake_images, _ = stage1_gen.predict_on_batch([embedding_batch, z_noise2])

            # Save images
            for i, img in enumerate(fake_images[:10]):
              print("Saving image")
              save_rgb_img(img, "/content/drive/MyDrive/Finalyearproject/Datasets/results_3/gen_{}_{}.png".format(epoch, i))
            stage1_gen.save_weights('/content/drive/MyDrive/Finalyearproject/Datasets/results_3/part1/stage1_generator.h5')
            stage1_dis.save_weights('/content/drive/MyDrive/Finalyearproject/Datasets/results_3/part1/stage1_discrimator.h5')
            ca_model.save_weights('/content/drive/MyDrive/Finalyearproject/Datasets/results_3/part1/stage1_ca.h5')
            embedding_compressor_model.save_weights('/content/drive/MyDrive/Finalyearproject/Datasets/results_3/part1/embedding_compressor.h5')
            adversarial_model.save_weights('/content/drive/MyDrive/Finalyearproject/Datasets/results_3/part1/adversarial_model.h5')
```

Fig 5.8

CHAPTER 7: OUTPUT

Generated image

Fig 7.1

Fig 7.2

Fig 7.3

CHAPTER 8:
IMPACT OF OUR PROJECT TOWARDS SOCIETY

Positive Impacts:

- Creative Industries: GANs have revolutionized creative industries such as art, design, and entertainment. They enable the generation of new and diverse visual and audio content, empowering artists and designers with novel tools for creative expression.
- Content Generation and Personalization: GANs have the potential to automate content generation across various domains. They can generate personalized recommendations, advertisements, and news articles, enhancing user experiences and engagement.

Negative Impacts:

- Data Privacy and Security: GANs also raise concerns regarding data privacy and security. The training of GANs often relies on large datasets, potentially containing sensitive information. Safeguarding data and ensuring responsible data handling practices are crucial to protect individuals' privacy and prevent unauthorized use.
- Deepfakes and Misinformation: GANs can be used to create realistic deepfake content, including manipulated images, videos, and audio recordings. This raises concerns about the spread of misinformation, fake news, and the potential for malicious use, such as impersonation or defamation.

As with any transformative technology, there are both positive and negative implications of GANs on society. It is important to strike a balance by promoting responsible use, addressing ethical considerations, and establishing regulatory frameworks to maximize the benefits while minimizing potential risks.

CHAPTER 9:

CONCLUSION

To tackle the challenging task of generating realistic high-resolution images, researchers introduced Stacked Generative Adversarial Networks (StackGAN-v1 and StackGAN-v2). These architectures aim to break down the complex problem into more manageable sub-problems. StackGAN-v1, enhanced with conditioning extensions, initially focused on text-to-image synthesis and introduced a novel sketch refinement process. By utilizing this approach, it becomes feasible to generate highly detailed images that closely match the provided textual descriptions.