Ministry of Higher Education
and Scientific Research
*** * ***
Carthage University
*** * ***
National Institut of Applied Sciences and Technologie

المعهد الوطني للعلوم التطبيقية و التكنولوجيا
Institut National des Sciences Appliquées et de Technologie

End of year project

Field : Automatic and Industrial IT

Level : 4Th Year

Subject:

Generating Human Faces based on Features

Prepared By :

KAROUI Mohamed Amine

BEN RAYANA Doua

FARHAT May

ACHICH Mohamed

Academinc Year: 2022/2023

Appreciation

It is with great pleasure that we reserve these lines as a sign of gratitude and recognition to Mr. HAMDI Mohamed Ali, who kindly provided us with proper guidance and ensured the most favorable conditions to carry out this work. We would like to express our heartfelt thanks for his

effective contribution, valuable collaboration, and fruitful advice that assisted us in accomplishing this work Finally, we would like to thank all those who have followed the tribulations that occurred

during this project, to our families and friends. May the members of the jury find our sincere

gratitudes for the honor they give us by accepting to validate this work and we are thankful for their evaluations.

Contents

General Introduction **1**

1 CelebA DataSet **2**
1.1 Introduction . . . 2
1.2 Generalities . . . 2
1.3 Data Processing . . . 3
1.3.1 Data Cleaning . . . 3
1.3.2 Data Augmentation . . . 4

2 Study in depth of Generative Adversarial Network **5**
2.1 Introduction . . . 5
2.2 GANs Architecture . . . 5
2.3 Mathematical Approach . . . 6
2.3.1 MiniMax Game . . . 6
2.3.2 Math behind GANs . . . 6

3 Implementation **8**
3.1 Introduction . . . 8
3.2 Implementation tools . . . 8
3.2.1 Environment . . . 8
3.2.2 Last Dataset preparation step . . . 8
3.3 Implementing the Generative Adversarial Networks . . . 9
3.3.1 Constructing the Discriminator Model: . . . 10
3.3.2 Constructing the Generator Model: . . . 11
3.4 Model Training . . . 12
3.4.1 Model optimization . . . 12
3.4.2 Model compilation . . . 12
3.4.3 Training time . . . 13

4 Deploy the model on Google Cloud Services and create a User Interface **14**
4.1 Introduction . . . 14
4.2 Web Application Overview . . . 14
4.2.1 Web Application system interactions . . . 14
4.3 Methodology . . . 15
4.3.1 Deploying GANs on Google Cloud Functions . . . 15
4.3.2 implementation of the web interface . . . 16
4.4 Conclusion . . . 16

5 The applications of face generators **17**
5.1 Introduction 17
5.2 Data augmentation 17
5.3 Industry 19
5.4 The future of face generator 20
5.5 Conclusion 20

General Conclusion **21**

List of Figures

1.1 Sample images with Bangs and Eyeglasses [4]. 3
1.2 Sample images with Wavy hair and Mustache [3]. 3

2.1 GAN Simplified Architecture [3]. 5
2.2 GAN's Loss Functions [3]. 6
2.3 GAN's Mathematical View [3]. 7

3.1 Diagram of GAN 9
3.2 Layers of GAN 10
3.3 LeakyRelU vs RelU 10
3.4 Transposed Convolution 11
3.5 Tanh function 11
3.6 Adam Optimizer 12
3.7 HyperParameters 13

4.1 System Architecture 15
4.2 Cloud Function Code 15
4.3 Website home page 16

5.1 Dataset Expansion 18
5.2 Generated Faces 19

General Introduction

In the fields of computer vision and artificial intelligence, GANs have made impressive strides toward producing high-quality images, including faces.

The goal of this project is to create a website that generates realistic and varied human faces using generative adversarial networks (GANs) in order to exploit its capabilities to give users an engaging and participatory experience.

The GAN model learns to produce artificial faces that are similar to real ones by using a generator and discriminator network in an adversarial manner. Age, ethnicity, hairdo, and facial expressions are just a few of the characteristics that the created faces display, creating a variety of visually appealing images. While, the web page acts as a demonstration for the GAN model's abilities after training. Users can investigate and engage with the generated faces to see the variety and excellence of the created synthetic images. It provides users with an interesting opportunity to see the possibilities of generative models in producing realistic human faces.

The project not only covers the generation of faces but also the difficulties in implementing the GAN model on the website. Considerations including model size, computing resources, and user experience are taken into account during optimizations to guarantee that the model is executed efficiently in the online context. Security measures are put in place to safeguard user data and offer a secure browsing environment.

Overall, this project blends web development with the state-of-the-art GAN technology to provide an aesthetically attractive and dynamic website. The project is to promote interest and awareness of the possibilities of artificial intelligence in the area of computer vision by giving users the chance to investigate and appreciate the power of generative models in producing human faces.

Chapter 1

CelebA DataSet

1.1 Introduction

In this chapter we will be discussing the content of our DataSet used in this project which is CelebA [2]. First of all we will be discussing its content and why it is widely used in the field of computer vision particularly when it comes to face recognition and face generation. Furthermore, we will be detailing its composition, annotation and how we managed to perform the data augmentation and the cleaning process.

1.2 Generalities

CelebA which stands for CelebFaces Attributes is a large scale dataset that's used in multiple fields such as Image recognition and Face generation. It contains more than 200k images(202599 Images) to be exact. The celebA dataset offers large pose variations with different backgrounds.

It has a large quantities and rich annotations including **10,177** number of identities and **40** attribute annotation

When it comes to the **Annotation** of images, each image in the CelebA dataset is annotated with 40 attributes including, hair color, gender, eyeglasses and more. These informations will provide a strong ground truth information that will be valuable for training and testing.

The figures 1.1 and 1.2 down below shows the different samples with multiple attributes.

Figure 1.1: Sample images with Bangs and Eyeglasses [4].

Figure 1.2: Sample images with Wavy hair and Mustache [3].

1.3 Data Processing

In this section we will be detailing the process of cleaning, transforming and augmenting our dataset in order to get better results in the output. Data Processing is a crucial step that must be performed before feeding the data to the model. It takes into consideration various activities that aims to manipulate raw data into a more meaningful and usable format.

1.3.1 Data Cleaning

Data Cleaning refers to the process of identifying and correcting anomalies in the Dataset in order to improve the accuracy of the model and prevents error from surfacing all over the code.

The Dataset contains some anomalies that are essentially divided into two categories:

- Missing values
- Duplicated samples

The **missing values** is identified while checking the names of different photos. The standard format for the photos is "xxxxxx" where each x refers to a number starting from "000001" to "202599". Yet some data samples presented a missing digit in the standard format.

The solution of this problem was to loop through all the data and use the **rename()** function of openCV in order to apply the right format for every sample.

Duplicate samples is also a problem to take into consideration, since it may skew the statistical analysis or modeling results. Also, in order to avoid over-fitting the model we mus eliminate all the duplicate samples form the data set.

In order to solve this issue, the data must be read in its shape via **image.shape()** function in openCv in order to compare it to each other and with the **PyTorch** library, removing duplicates can be done.

1.3.2 Data Augmentation

Data Augmentation is a technique used in order ti improve the size of our dataset by increasing the number of samples. Multiple techniques could be used in order to enhance the dataset.

Different ways are used in this project in order to do the data augmentation.

- Image Rotation and Flipping: rotate the images according to different angles, also, flipping them in different positions: Horizontally and Vertically
- Data Wrapping: applying geometric deformations such as stretching and twisting the images.
- Image Blurring: Adjusting the images with different bur filters such as the Gaussian blur and Motion blur.

After applying those techniques to our Dataset, the results of cleaning and augmentation are written in the table down below.

Table 1.1: DataSet Size

Original Dataset	Cleaned Dataset	Augmented Dataset
202599	201326	202235

Chapter 2

Study in depth of Generative Adversarial Network

2.1 Introduction

Generative Adversarial Network also known as GANs is a powerful framework that uses a different approach in the Deep Learning Field. Its architecture is quiet different since it offers the ability to learn via adversarial network compositions and helps generates realistic data samples. In this chapter we will be detailing more about its composition, the mathematical theories behind and the full training process.

2.2 GANs Architecture

When it comes to the GANs core, it is basically composed by two main components the Generator and the Discriminator. It learns the dataset distribution by pulling those two neuronal networks against each other.

The figure 2.1 down below gives a general overview of GANs structure.

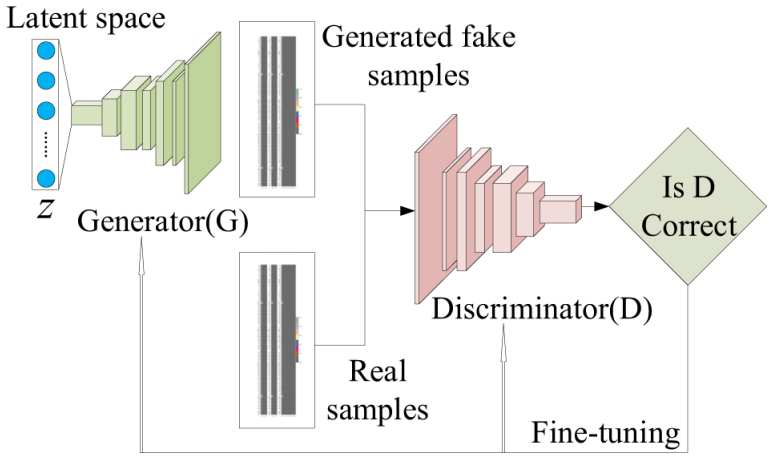

Figure 2.1: GAN Simplified Architecture [3].

Before diving more into the components of GAN, let's understand how it functions in general. By adding some random noise and implement it to the Generator architecture, the generator will try to generate synthetic data samples which means that the data generated artificially that does not exists in the real dataset that was fed to the network. As for the Discriminator, its job refers to learn how to distinguish between real and generated Sample.

In conclusion to what have been said, the goal of the **Generator** is to generate fake images in order to misconduct the work of the discriminator. While the goal of the **Discriminator** is to identify the real from the fake data.

2.3 Mathematical Approach

2.3.1 MiniMax Game

MinMax refers to an optimization strategy, it is specifically used while working with Generative Adversarial Network. To explain it more, it is an optimization algorithm while working in a turn-based game between two players. To main goal here is to minimize the worst case potential loss. In other words, a player must consider very well all the best strategies and counter the movements of its opponent by identifying the best response to it.

If we apply this strategy to the GAN's model, we can simply put that the **Generator** and the **Discriminator** are the two players who are taking turns and instantly updating their model weights. The idea here resumes in the minimization of the **Generator's** loss function and maximizing the **Discriminator's** loss function. More mathematical theories will be explained in the next section.

2.3.2 Math behind GANs

Every Neuronal network should be identified by its Loss function. The Loss or Cost function is a mathematical function that measures the disagreement between the predicted output and the essential target. By computing the loss we will be able to tune its parameters in order to get the closes output to the ground truth.

Now let's take a look at both Loss Functions which will be displayed in the figure down below 2.2

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$
$$J^{(G)} = -J^{(D)}$$

Figure 2.2: GAN's Loss Functions [3].

Let's dive deeper in the components of those functions:

- The first Term in J(D) refers to feeding the actual Data to the discriminator.
- The second Term in J(D) is actually the Fake Data samples generated by the Generator.
- The J(G) stands for the Generator Loss Function.

The Discriminator role stands in finding the best approach while learning to maximize the average of the log probability pf real images and the log of the inverse probability for the fake images which stands for this part of equation $\mathbf{log\ D(x)} + \mathbf{log(1 - D(G(z)))}$.

On the other hand, the Generator will try intensively to minimize the inverse probability predicted by the Discriminator for fake images. This will give an encouraging push for the Generator to generate samples with a low probability of being fake. So, technically the Generator will try to minimize this equation $\mathbf{log(1 - D(G(z)))}$.

The image 2.3 down below details all those functionalities that were mentioned earlier.

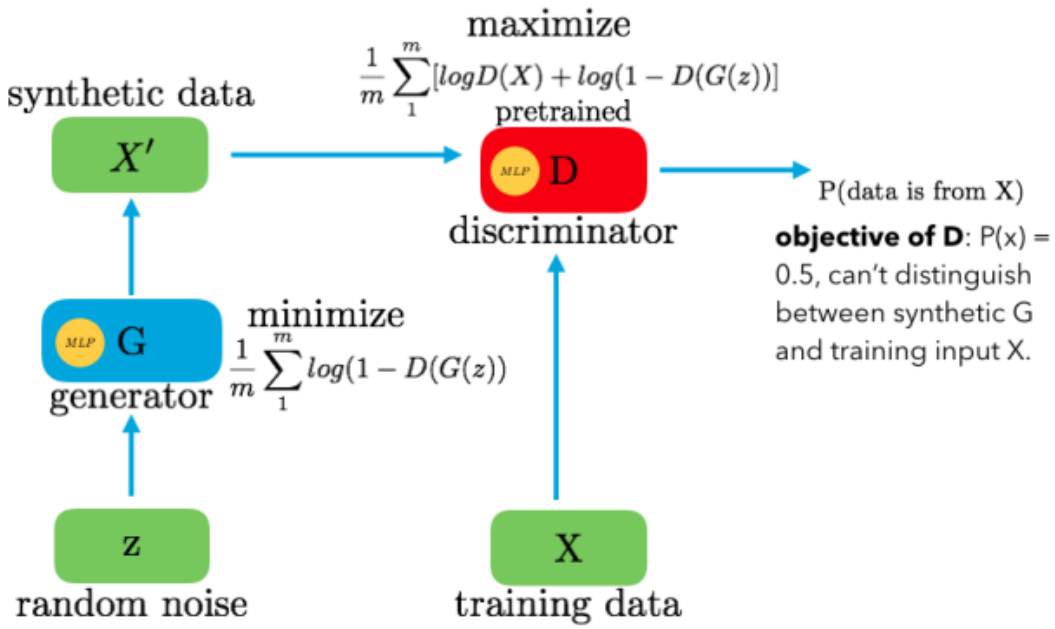

Figure 2.3: GAN's Mathematical View [3].

Chapter 3

Implementation

3.1 Introduction

This chapter is about the implementation of a Generative Adversarial Network (GAN) for the interesting challenge of face generation. GANs have transformed artificial intelligence by allowing the development of incredibly realistic and innovative pictures. GANs can learn to construct convincing faces with attributes comparable to genuine human faces by combining a generator and a discriminator network.

3.2 Implementation tools

3.2.1 Environment

The advancement of machine learning libraries that are accessible and easy to import has contributed to the success of the field of artificial intelligence in general, and Deep Learning in particular.

- Jupyter Notebook : is an open-source web-based interactive computing environment used for data analysis, visualization, and prototyping. It allows users to write and execute code in cells, making it ideal for iterative development and exploratory data analysis. Notebooks can be saved, exported, and shared, facilitating collaboration and reproducible.
- Keras : a popular, high-level neural network API written in Python that runs on TensorFlow or Theano. It prioritizes fast experimentation and prototyping of deep learning models. Keras serves as an intuitive interface, allowing users to easily configure and build neural networks independently of the underlying library. For face generation projects with GANs, Keras simplifies the process by providing a user-friendly syntax and seamless integration with powerful deep learning frameworks.

3.2.2 Last Dataset preparation step

We will be working with the CelebA Dataset as mentioned earlier, which contains over 200,000 celebrity photos with annotations. However, for face generation, we will solely use the photographs themselves, without any annotations. The images in the collection are all color, with three color channels (RGB).

We have conducted some pre-processing procedures for you in order to simplify the process and focus on developing the GANs. The CelebA photos were cropped to eliminate non-facial areas and scaled to 64x64 pixels, resulting in a pre-processed dataset that is a subset of the original CelebA data.
We used with that a *DataLoader* :
The DataLoader is in charge of loading and processing the dataset for training the GAN model in an effective manner. It handles duties like data shuffle, image batching, and enabling quick access to data during training. We may manage the quantity of data processed at each iteration and the resolution of the images used for training by setting the batch size and image size.

3.3 Implementing the Generative Adversarial Networks

With the diagram below 3.1, you can find out about the architecture and potential of Generative Adversarial Networks (GANs)

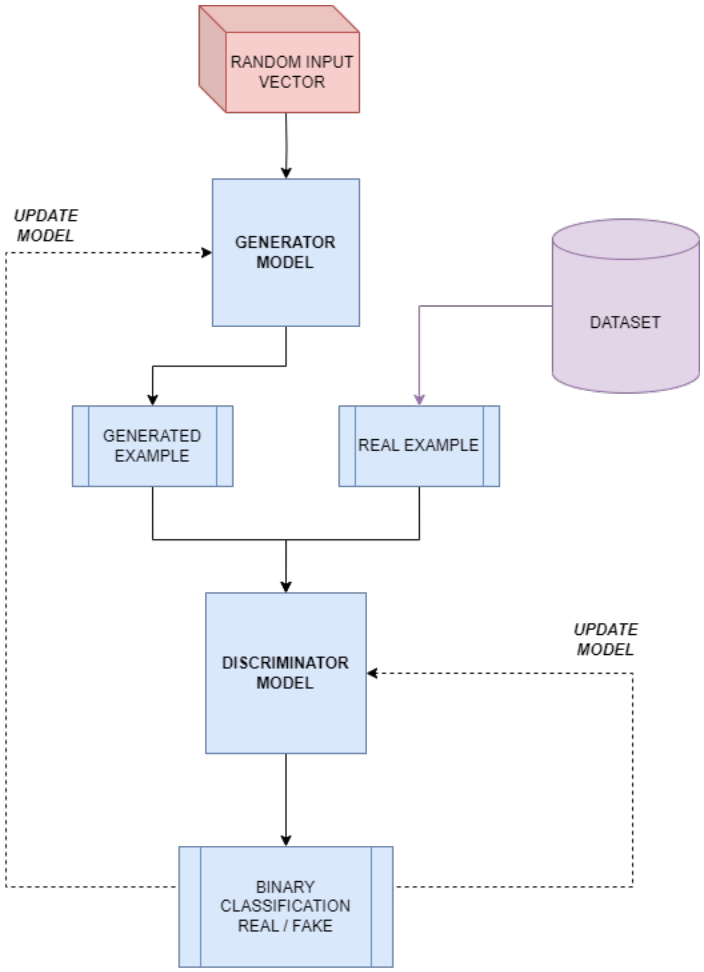

Figure 3.1: Diagram of GAN

3.3.1 Constructing the Discriminator Model:

we will forge the discriminator model tailored for the generative network.
There are several important components and concepts:

- Convolutional Layers 3.2: The discriminator is made up of numerous convolutional layers that extract features from the input pictures. Convolutions aid in the acquisition of spatial information and data patterns.

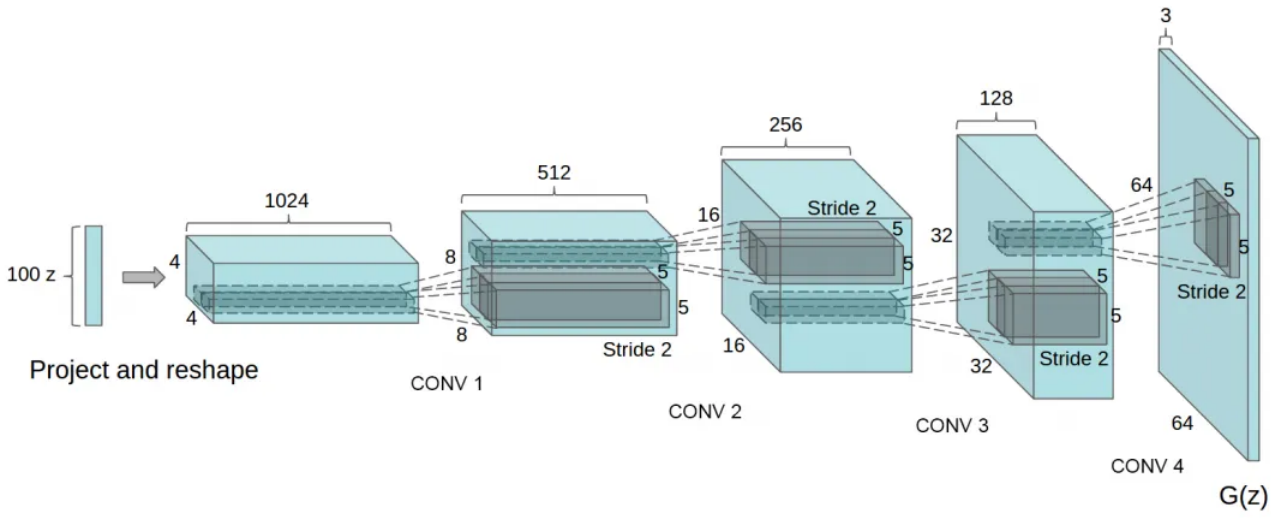

Figure 3.2: Layers of GAN

- Leaky ReLU Activation(Rectified Linear Unit) 3.3: To induce non-linearity, the code use the Leaky ReLU activation function because it prevents "dying" ReLU units by permitting small negative values, which might be useful for training deep networks.

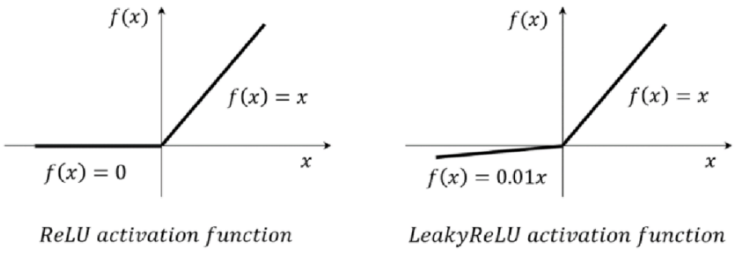

Figure 3.3: LeakyRelU vs RelU

- Batch Normalization: Except for the first convolutional layer, batch normalization is conducted after each convolutional layer. It normalizes the activations within each mini-batch, which can speed up training and improve generalization.

- Helper Function: The conv function is a helper function that constructs a convolutional layer with batch normalization as an option. By encapsulating the steps required, it streamlines the process of creating convolutional layers.

3.3.2 Constructing the Generator Model:

- Transpose Convolutional Layers 3.4: Transpose convolutional layers are used by the generator to upsample the input latent vector and build a new picture of the same size as the training data (32x32x3). Convolutions that transpose assist to enhance the spatial dimensions of the input.

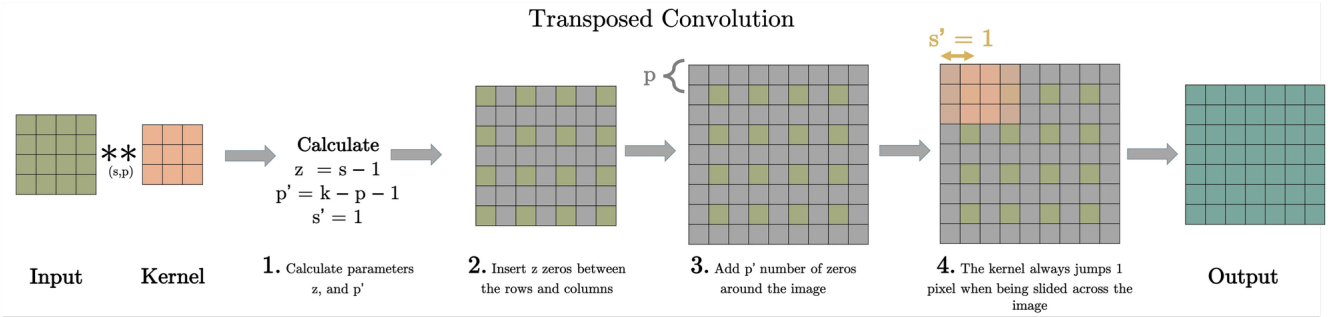

Figure 3.4: Transposed Convolution

- Helper Function: The deconv function is a helper function that cgenerates a transpose-convolutional layer with batch normalization as an option. By encapsulating the necessary methods, it simplifies the process of defining transpose convolutional layers.
- **ReLU** and **Tanh** Activation 3.5: Except for the last transpose convolutional layer, ReLU activation is performed after each transpose convolutional layer. It adds nonlinearity and aids in the elimination of vanishing gradients during training. The last layer employs a Tanh activation to compress pixel values between -1 and 1, resulting in the final output picture.

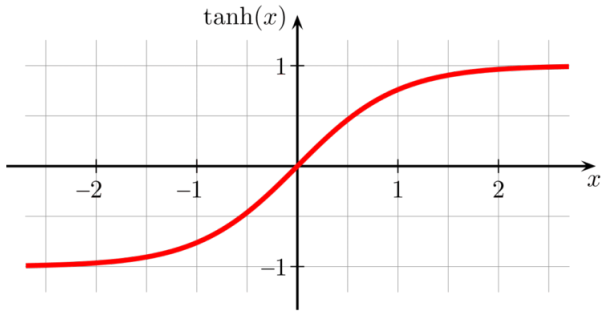

Figure 3.5: Tanh function

3.4 Model Training

3.4.1 Model optimization

In this project, we used one of the most famous optimizers in the domain of deep learning: The Adam Optimization Algorithm

The Adam optimizer is critical in the training of generative adversarial networks (GANs) for image-generating applications. The generator network in the context of GANs strives to create realistic pictures, whilst the discriminator network learns to discriminate between actual and fraudulent images. During the training phase, the Adam optimizer is in charge of updating each network's parameters depending on the computed gradients and the chosen learning rate. helping the convergence of the GAN model.
The GAN model benefits from the Adam optimizer's capacity to handle non-stationary objective functions and vast parameter spaces. The method changes the learning rate for each parameter adaptively, enabling efficient exploration. The addition of first- and second-moment estimates reduces the influence of noisy gradients, resulting in more stable and trustworthy updates. **NB:** We tried different optimizers like Adagrad and RMSProp before settling down to Adam.

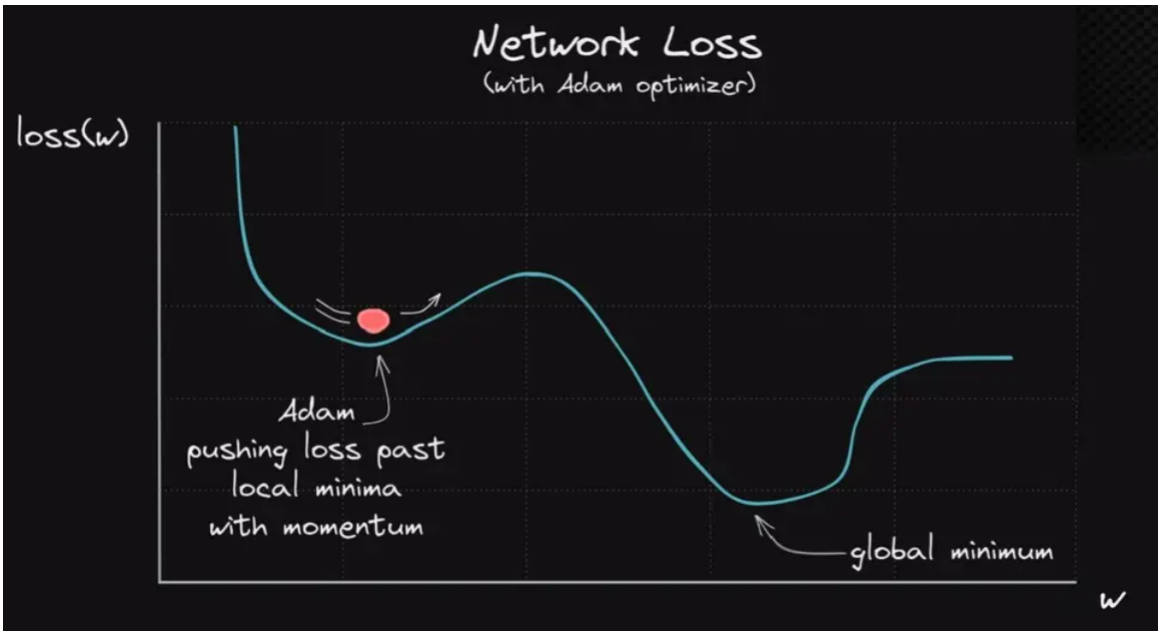

Figure 3.6: Adam Optimizer

3.4.2 Model compilation

The discriminator and generator networks of the described GAN architecture are constructed with specific parameters to allow the making of realistic pictures.

The discriminator network uses a sequence of convolutional layers, batch normalization, and a fully connected layer to determine the validity of input pictures. To introduce non-linearity, it utilizes a LeakyReLU activation function.

The generator network, on the other hand, takes a random noise vector as input and employs a combination of fully connected and transposed convolutional layers to progressively upsample the input and produce high-quality pictures. Except for the last transposed convolutional layer that provides the final RGB picture, batch normalization is done to each layer of the generating network.

```
# Create discriminator and generator
D, G = build_network(d_conv_dim, g_conv_dim, z_size)
```

```
Discriminator(
  (conv1): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  )
  (conv2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (conv3): Sequential(
    (0): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (fc): Linear(in_features=4096, out_features=1, bias=True)
)

Generator(
  (fc): Linear(in_features=100, out_features=4096, bias=True)
  (t_conv1): Sequential(
    (0): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (t_conv2): Sequential(
    (0): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (t_conv3): Sequential(
    (0): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  )
)
```

Figure 3.7: HyperParameters

3.4.3 Training time

Due to the significant time required for image generation, the GAN model training procedure was noted to be slow. Creating high-quality images requires a massive amount of computational and processing capacity. Various ways were investigated to minimize this, including running the training on other platforms such as CPUs, GPUs accessible on Kaggle notebooks, and even external GPUs (using Kuda toolkit). By exploiting parallel computing capabilities and improving resource allocation, these techniques attempted to reduce training time. The quality of the produced pictures was shown to be greatly impacted by both the dataset used for training and the length of the training procedure. Longer training cycles allowed the model to understand complicated patterns and more successfully represent the underlying distribution of the data, resulting in better image quality. Thus, attaining excellent results in GAN-based image-generating tasks requires a careful balance of training length and dataset selection.

Chapter 4

Deploy the model on Google Cloud Services and create a User Interface

4.1 Introduction

This chapter describes the implementation of a web application that allows quality control engineers to interact with a Generative Adversarial Network (GAN) for face generation. The web application consists of a website that makes API calls to a GAN model deployed on Google Cloud Functions. This chapter is organized into two sections as follows: section 1 introduces the system architecture and interactions. Section 2 discusses the implementation details of the Web Application

4.2 Web Application Overview

Our web Application is divided into two main parts :

- Google Cloud Server-less back-end: Our AI model was deployed using Google server-less cloud function structure, a Google cloud service that provides dynamic resource allocation based on the traffic demand. This means that our application can scale up or down automatically, even to zero if there is no traffic. This feature makes it convenient to manage and suitable for this project.
- Front-end website: this is the part of the web application that users interact with on their browsers. It is built with NextJs a JavaScript library that lets you create user interfaces with reusable components: pieces of code that can display data and handle interactions.

4.2.1 Web Application system interactions

When a user clicks a button on the website, it triggers an HTTP request that activates the cloud function. This cloud function creates a new instance of the (GAN) model, which then takes random noise as input and generates an image. The image is returned to the website as an HTTP response and displayed to the user, as shown in the diagram below 4.1 :

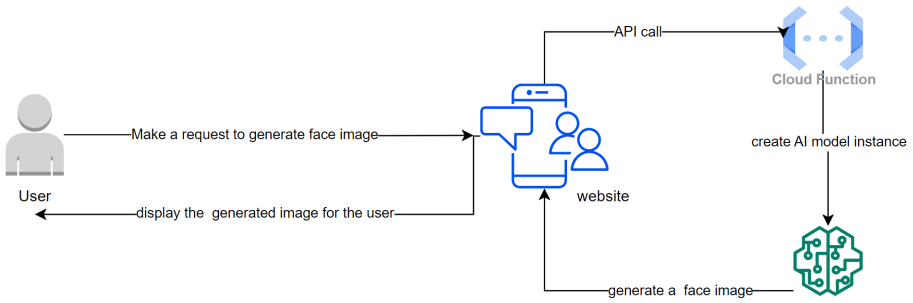

Figure 4.1: System Architecture

4.3 Methodology

4.3.1 Deploying GANs on Google Cloud Functions

in this section, we demonstrate the process of saving a trained Generative Adversarial Network (GAN) model using PyTorch and deploying it on a Google Cloud server-less function.

- Saving the GAN model: After training our GAN model using PyTorch, we saved the model's state dictionary using the torch.save() function.
- Creating a Google Cloud Function: We navigated to the Cloud Functions page in the Google Cloud Console and created a new function. We chose an appropriate name, memory allocation, and a http trigger for our function.
- Uploading the saved model to Google Cloud Storage: We uploaded our saved PyTorch model file to a Google Cloud Storage bucket. This allowed us to access the model from our Google Cloud Function.
- Writing the function code: In the inline code editor, we wrote the code for our function. This included loading our saved PyTorch model and using it to generate an image on request.

```
import torch
from google.cloud import storage

# these values are masked for security
BUCKET_NAME = ******************************
MODEL_FILE_NAME = ***********************************

def generate_image(request):
    # Load the saved PyTorch model from the google storage
    client = storage.Client()
    bucket = client.get_bucket(BUCKET_NAME)
    blob = bucket.get_blob(MODEL_FILE_NAME)
    model = torch.load(blob.download_as_string())

    # Generate an image using the model
    image = model.generate_image()

    # Return the generated image as an HTTP response
    return image
```

Figure 4.2: Cloud Function Code

- Deploying the function: Once we had written the code and uploaded our saved model, we deployed the function by clicking the "Deploy" button.
- Testing the function: After deploying the function, we tested it by sending a request to the cloud function http endpoint. The function returned an image generated by our GAN model.

4.3.2 implementation of the web interface

we created a NextJs component that handles the button click event and sends an HTTP request to the cloud function endpoint using the fetch API. The component also includes logic to handle the response from the cloud function and update the website's state to display the generated image.

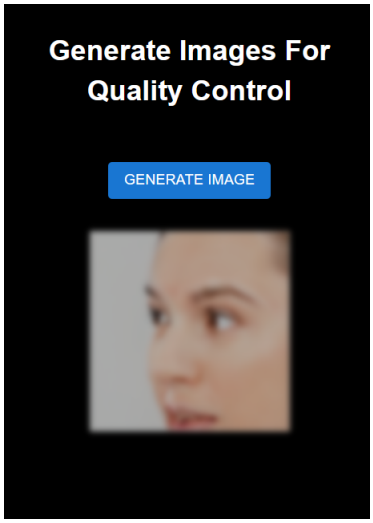

Figure 4.3: Website home page

4.4 Conclusion

This chapter presented the implementation of a web application that allows users to generate realistic faces using a GAN model deployed on Google Cloud Functions. The web application consists of a front-end website built with NextJs and a back-end cloud function that runs the GAN model on demand. The web application demonstrates how to use server-less cloud computing services to deploy AI models in a scalable and cost-effective way. The web application also provides a user-friendly interface for quality control engineers to interact with the GAN model.

Chapter 5

The applications of face generators

5.1 Introduction

This chapter explores the numerous uses of face generators in a variety of domains, with a focus on their importance in data augmentation. Face generators are able to create incredibly realistic faces that closely mimic real human characteristics by utilizing cutting-edge technology, greatly advancing artificial intelligence.

This chapter examines how face generators play a crucial role in strengthening and increasing training datasets, which eventually enhances the performance of facial analysis tasks like age estimation, emotion detection, and recognition. Face generators have applications in a wide range of fields, and we illustrate their potential to change a number of industries throughout the chapter.

5.2 Data augmentation

Through the creation of artificial face images, face generators significantly contribute to the growth and diversification of training datasets. These produced photos make a substantial contribution to data augmentation, a technique that improves the efficiency of facial analysis jobs. Face generators add a variety of facial variations and features, such as age, gender, ethnicity, and expressions, by integrating synthetic face photos into the training dataset.

The limits brought on by a lack of real-world data are solved through data augmentation utilizing synthetic facial photos. It expands the variety and volume of training data, enabling machine learning models to gain knowledge from a wider range of examples. As a result, the model's capacity to generalize, accurately identify, and interpret faces in realistic situations is enhanced.

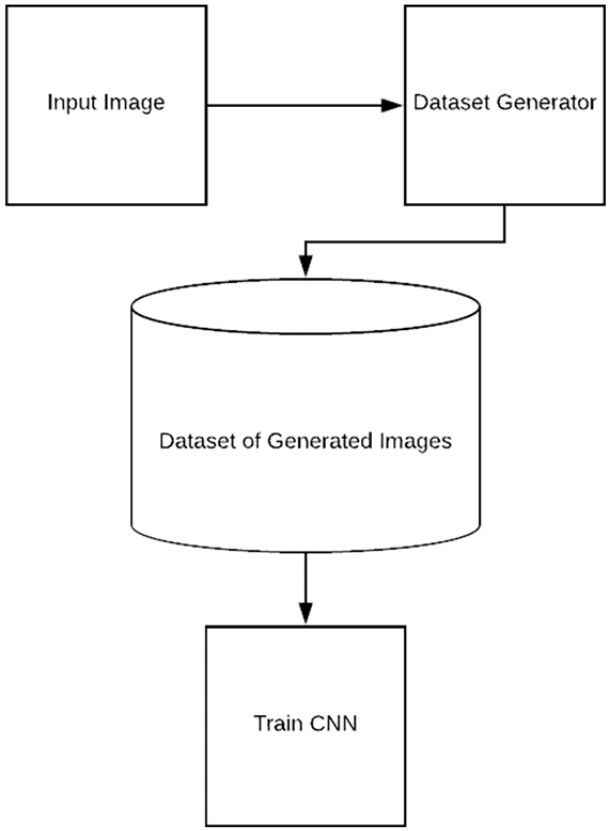

Figure 5.1: Dataset Expansion

There are many advantages to data augmentation in facial analysis applications.

First off, it aids in preventing overfitting, a problem where a model becomes overly focused on training data and struggles to generalize effectively to new data. Data augmentation allows the model to acquire robust and discriminative features by supplying synthetic face photos with a variety of properties, lowering overfitting and enhancing generalization performance.

Second, data augmentation aids in addressing the problem of unbalanced datasets. Certain variables, such as particular age groups or races, may be underrepresented in real-world datasets, which might skew model projections. Data augmentation can help to reduce bias and provide fair and objective facial analysis by creating synthetic face photos with balanced attribute distributions.

Additionally, models can learn and detect different facial expressions thanks to data augmentation utilizing face generators, which is essential for tasks like emotion analysis and sentiment recognition. Models can learn the tiny differences in facial features that correspond to various emotions by observing synthetic face images with various expressions, which enhances their performance in these tasks.

5.3 Industry

Face generators have diverse applications in numerous types of fields, demonstrating their adaptability and influence on multiple industries.

- **Privacy Protection:** By creating artificial faces that closely resemble real faces without disclosing the identities of people, face generators can contribute to privacy protection. This method can be used in situations when protecting privacy is important, including in surveillance systems used by the general public or when exchanging data for research. Organizations can anonymize private facial data while preserving the data's value for analysis and research by employing face generators.

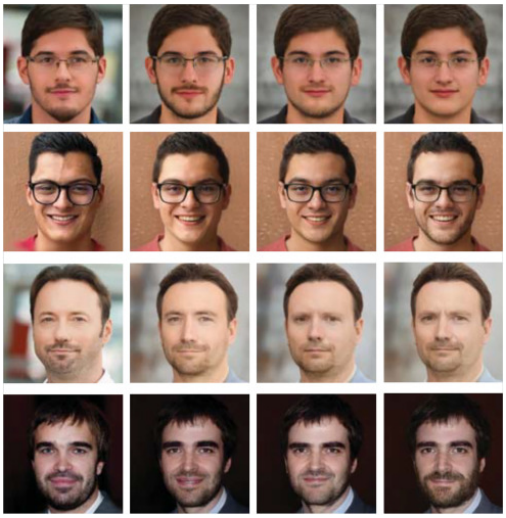

Figure 5.2: Generated Faces

- **Entertainment:** Face generators allow the production of incredibly lifelike computer-generated characters for motion pictures, video games, and virtual reality experiences. With the help of these generators, designers and artists may create characters with a wide variety of facial expressions and traits, which improves storytelling and immersive entertainment.
- **Beauty and Fashion:** Face generators are crucial for allowing virtual try-on experiences. Without physically applying anything, users can see how various cosmetic looks, hairstyles, or virtual accessories would appear on their own faces. By enabling customers to experiment with various appearances and make more educated purchasing decisions, our application improves the customer experience.
- **Art and design** Face generators encourage artistic innovation by giving creators of characters, clothing, and visual art access to a wide variety of facial traits. Face generators can be used by artists to experiment with distinctive and varied facial aesthetics, stretching the bounds of originality and expression in their work.

As face generator technology develops, it has the potential to change a number of industries by empowering people's decision-making and creative processes by providing them with lifelike synthetic faces.

5.4 The future of face generator

Face generators have a huge amount of promise for future development and innovation. Face generators should produce even more lifelike and high-fidelity synthetic faces as technology advances. The accuracy, diversity, and quality of the generated faces can be improved using advanced algorithms and deep learning approaches, pushing the limits of realism.

The fields of virtual reality and augmented reality constitute one area of future growth. Face generators can be crucial in the creation of incredibly lifelike avatars and digital portraits of people for immersive VR/AR experiences. This could result in more interesting narratives, improved social interactions, and tailored virtual environments.

Additionally, face generators might benefit medical and healthcare applications. By creating lifelike virtual patients, they can help in medical simulation and training, enabling medical personnel to practice difficult procedures and hone their abilities in a secure setting. Face generators can also help with medical diagnosis by analyzing and identifying facial patterns linked to various diseases and ailments.

Furthermore, improvements in face generator technology have the potential to completely change the robotics industry. These generators can improve human-robot interactions by producing incredibly lifelike and expressive robot faces. This makes robots more personable, sympathetic, and efficient in helping people with a variety of jobs.

Face generators have a wide range of possible uses in industries like education, gaming, advertising, and more. Face generators have the potential to alter our digital experiences by bridging the gap between the virtual and real worlds with further research and development.

5.5 Conclusion

In conclusion, face generators have become effective tools with a wide range of uses. They enhance virtual experiences, encourage artistic expression, and protect privacy by creating artificial faces that maintain anonymity. Face generator technology has a bright future and has the ability to grow significantly, spurring innovation across many industries and creating new opportunities. Face generators are positioned to play a significant role in influencing the future of facial analysis, digital experiences, and artistic efforts as technology advances.

General Conclusion

The objective of this project is to generate fake faces and upload the model to a website that we created. Certainly, it was an exciting endeavors and a strong challenge to our capabilities.

In summary, GANs has shown an amazing capabilities in generating non-realistic and high-quality images that includes human faces. This step is a revolutionary one since it helps in multiple fields such as augmenting data when it comes to face recognition algorithms and enhance their performances.

In addition to that, GANs shows the ability to analyse multiple details and variations in facial features that helps generate visually appealing faces that approaches the ground truth very much.

Also, the web deployment part is intriguing as we should take into consideration the computational requirements of running a model on a website and optimize it accordingly

However, this project presents multiple challenges in different terms such as the Training Time and Resources. In fact, training GANs with high resolution images is essentially time consuming and requires high level of GPUs and large amount of memory. Also GANs can suffer from mode collapsing which means that the Generator fails to capture the full diversity of data and this can result in generating faces that are similar to the images provided within the data-set.

To successfully address those challenges, we required high knowledge in both GANs and optimization processes in Web Development. Yet , this project provides a useful hand in the industry field and importantly in quality control as it offers the chance for the production team to test their products on real-faces in order to conduct a full user-experience market study.

Bibliography

[1] Adam: A method for stochastic optimization. URL: https://arxiv.org/pdf/1412.6980.pdf.

[2] Celeba dataset downloads. URL: https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html.

[3] Dataset annotation. URL: https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html.

[4] Dataset processing. URL: https://www.altexsoft.com/blog/datascience/preparing-your-dataset-for-machine-learning-8-basic-techniques-that-make-your-data-bet

[5] Face deidentification with controllable privacy protection. URL: https://www.sciencedirect.com/science/article/pii/S0262885623000525.

[6] The gan landscape: Losses, architectures, regularization, and normalization. URL: https://openreview.net/pdf?id=rkGG6s0qKQ.

[7] Gan's mathematical approach. URL: https://jonathan-hui.medium.com/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09.

[8] Gan's work explained. URL: https://www.kionetworks.com/en-us/blog/what-are-generative-adversarial-networks-gans.

[9] Image data augmentation for facial recognition. URL: https://manmeet3.medium.com/face-data-augmentation-techniques-ace9e8ddb030.

[10] Keras imagedatagenerator and data augmentation. URL: https://pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/.

[11] Kers documentation/api. URL: https://keras.io/api/.

[12] Learn more abou gans. URL: https://www.techtarget.com/searchenterpriseai/definition/generative-adversarial-network-GAN.

[13] A mathematical introduction to generative adversarial nets (gan). URL: https://arxiv.org/pdf/2009.00169.pdf.

[14] Pytorch documentation — pytorch 2.0 documentation. URL: https://pytorch.org/docs/stable/index.html.

[15] Transferable face image privacy protection. URL: https://link.springer.com/article/10.1007/s40747-021-00399-6.