Fake Face Generation using Generative Adversarial Networks (GAN)

Team: 06 Members: Anusha chilakamarri, Faisal Mohammed, Indhu sri Krishnaraj, Jitesh Singamsetty, Lakshmi Keerthi Gopireddy, Meghana Kesani, Ram srinivas katragadda

Problem: Generating realistic synthetic faces using Generative Adversarial Networks (GANs).

Problem Type: Optimization Problem

Objective: To develop a generative model capable of producing realistic, high-quality synthetic faces indistinguishable from real faces.

Description: The purpose of this project is to construct realistic synthetic faces using a Generative Adversarial Network (GAN). GANs are composed of two neural networks: a Generator and a Discriminator. The generator generates synthetic images, while the discriminator validates their authenticity by comparing them to real ones.

To achieve the objective, the Generative Adversarial Network will perform the following steps:

1. Assign random weights to the generator and discriminator networks
2. Using the generator, create a batch of synthetic faces
3. Train the discriminator to recognize the difference between genuine faces (from a dataset) and synthetic ones created by the generator
4. Based on the feedback from the discriminator, update the generator to generate more realistic faces

The purpose of this approach is to reduce the difference between the created synthetic faces and real faces, as evaluated by the discriminator's ability to accurately identify them. The generator and discriminator networks are always working to improve their respective performances, with the generator seeking to generate more realistic faces and the discriminator attempting to classify genuine faces and synthetic faces.

Therefore this is an optimization problem in which a Generative Adversarial Network aims to generate realistic synthetic faces by minimizing the loss function of the generator and maximizing the accuracy of the discriminator. The agent iteratively improves the generator and discriminator networks to achieve this goal.

Methodology

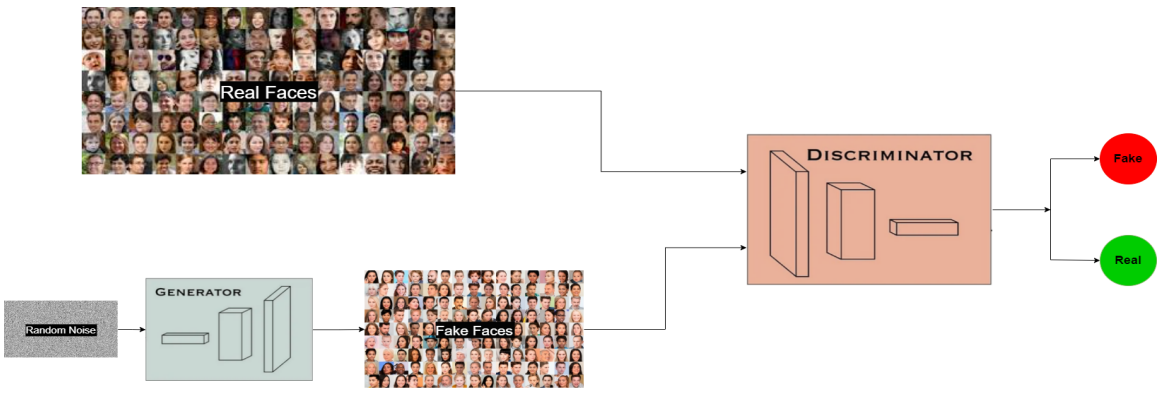

Fig1.1: Our Generative Adversarial Networks (GAN) Framework

Components:

- Data: To train the discriminator and guide the generator, a dataset of real-world face images is required. This dataset should include a diverse range of faces in order to generate faces to be diverse and realistic. In our previous checkpoint we have mentioned our dataset.
- Model: The GAN model includes two main parts: a generator and a discriminator. Both are neural networks, and typically they are implemented with deep learning methods such as convolutional neural networks (CNNs) for image data.
    - a. Generator: As an output, it generates fake face images from a random noise vector. The goal of a generator is to create images that are indistinguishable from real faces.
    - b. Discriminator: Takes both real and synthetic face images generated by the generator as input and outputs the probability that an image is real or synthetic. The discriminator is trained to classify real and synthetic images correctly.
- Objective: The goal is to optimize the generator and discriminator networks in a way that the generator generates a realistic synthetic face, and the discriminator cannot tell the difference between real and synthetic faces. The optimization is carried out using a loss function that compares real and synthetic faces.
- States: Currently the generator and discriminator is assigned random weights and once it learns from the images the weights keep on changing thus changing states. The generator has 12 convolution layers whereas the discriminator has 5 convolution layers.
- Environment: Working on without mask faces dataset downloaded from Kaggle, To implement and train the model we would be requiring high computational resources ie. GPUs. This will be achieved through Google Colab.
- Actions: The GAN model undergoes a training process that includes the following actions:
    - The generator takes a random noise vector as input and outputs a synthetic face image.
    - The discriminator evaluates the synthetic face image and returns a probability of whether it is real or fake.
    - The loss function computes the difference between the discriminator's output and the image's true label (real or synthetic).
    - To minimize the loss function, the parameters of both the generator and the discriminator are updated using backpropagation and gradient descent.
    - The process is repeated several times until the generator generates realistic synthetic faces.

Model Description:

There are two branches in the model:

Generator:

- Input: The generator takes a random noise vector (also known as a latent vector) as input. Typically, this latent vector has a fixed size, like 100 dimensions.

- Processing: The input is processed by the generator through a series of deconvolutional or transposed convolutional layers, with activation functions (like ReLU, Leaky ReLU) and normalization layers (like batch normalization) in between. The network selects the input gradually, increasing its spatial dimensions while decreasing its depth, until it is finally converted into an RGB image with the same dimensions as the actual faces in the dataset.
- Output: As a result of its operation, the generator creates a fake image of a face, which is then sent to the discriminator for analysis.

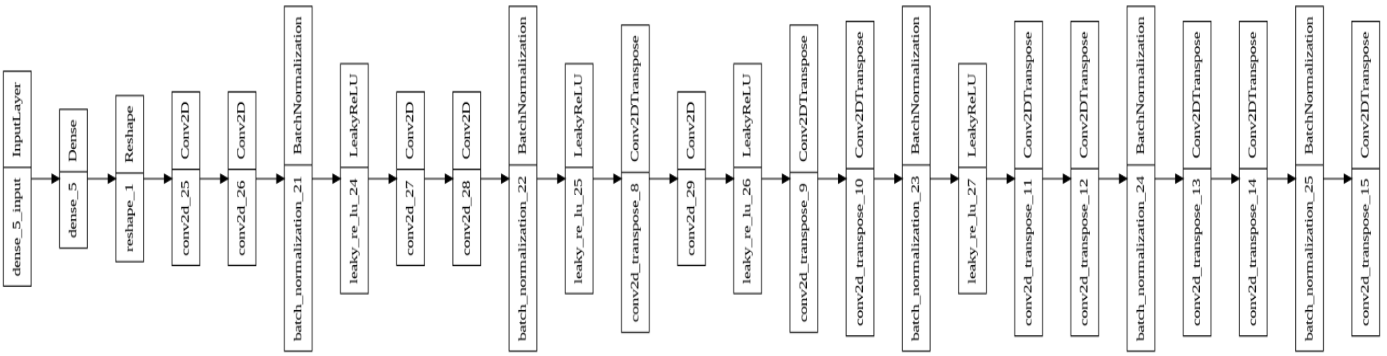

Fig 1.2: Generator Model Summary

Discriminator:

- Input: The discriminator is fed two different kinds of input: actual face images from the dataset and fake faces created by the generator. The size (e.g., 64x64 or 128x128 pixels) and color channels (e.g., RGB) of both types of images are the same.
- Processing: The discriminator runs the input images through a series of convolutional layers, interspersed with normalization layers (such as batch normalization) and activation functions (such as ReLU, Leaky ReLU). In order to extract features that allow the analyzer to distinguish between real and fake images, the layers progressively deepen the input and reduce the spatial dimensions. Using a fully connected layer and sigmoid activation on the output, the probability score is obtained.
- Output: A probability value between 0 and 1 generated by the discriminator expresses the probability that the input image is real. A value near 1 denotes a high likelihood that the image is real, while a value near 0 denotes a high likelihood that the image is fake.

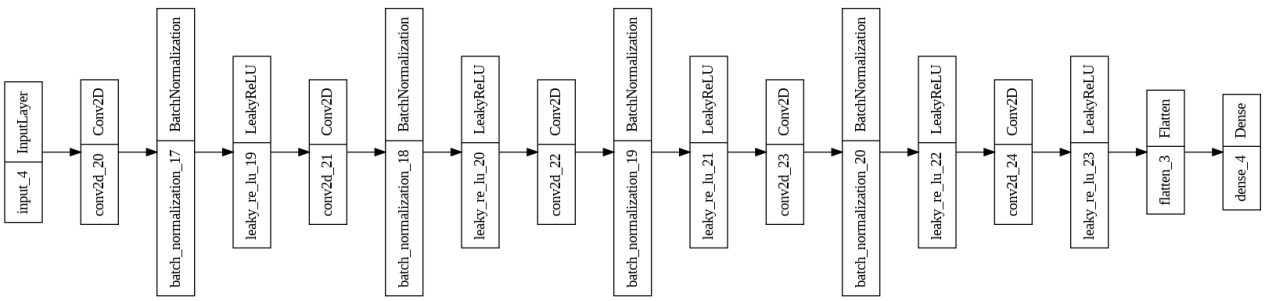

Fig 1.3: Discriminator Model Summary

In an adversarial training process, the discriminator and generator are trained simultaneously. The discriminator's goal is to correctly distinguish between real and synthetic images, whereas the generator's goal is to create synthetic images that the discriminator cannot tell apart from real images. The ability of the generator to create more realistic faces and the discriminator to distinguish between real and synthetic faces both get better with training. Until the generated synthetic faces are almost different from real faces, the discriminator cannot correctly classify them.

Implementation :

- Programming languages: We would be implementing the GAN framework in Python using deep learning libraries such as TensorFlow.
- Platforms: The model will be trained on Google Colab for training with the help of GPUs or TPUs.
- Core Algorithm/Pseudo Code: Initialize the generator and discriminator networks with random weights. Repeat until convergence, then generate a batch of synthetic faces using the generator. Train the discriminator on a mix of real faces and synthetic faces. Update the generator based on the feedback from the discriminator.
- Choice of Hyperparameters: For now we have taken RMSprop as the optimizer with learning rate 0.0001,Binary CrossEntropy is used as loss function. Epochs are 5 in the start and will change later on how the model behaves. Batch size is taken to be 32. These hyperparameters can be tuned using techniques like grid search, random search, or Bayesian optimization.
- Pre-processing Steps: Pre-processing of the dataset's images may be necessary to increase the diversity of the training data, such as resizing to a uniform size, normalization.It is performed to properly input data in the model.

Evaluation :

Several quantitative and qualitative metrics can be used to evaluate the efficacy of the proposed GAN approach for creating synthetic faces.

Quantitative Metrics:

Frechet Inception Distance (FID): FID assesses the statistical similarity between actual and synthetic images by comparing their feature distributions within the Inception Network. Lower FID ratings imply that the generated images resemble the actual images more closely, indicating improved performance.

Precision, Recall, and F1-Score: These metrics can be used to assess the equilibrium between image quality and variety. A high precision indicates that the generated images are of excellent quality, whilst a high recall suggests that the model creates a wide variety of images.

Qualitative Evaluation:

Visual Inspection: By visually inspecting a sample of created photos, one can evaluate the visual quality, realism, and diversity of the synthetic faces. Comparing the generated faces with genuine faces from the dataset can help uncover any potential artifacts or implausible characteristics.

Human Evaluation: Conducting a user study in which users are asked to identify between real and synthetic faces might shed light on the model's efficacy. A higher incidence of confusion between genuine and artificial faces is indicative of improved performance.

Challenges and Limitations:

Mode Collapse: In some instances, the generator may only produce a limited number of synthetic faces, resulting in a lack of diversity among the images generated. This condition, known as mode collapse, might result in undesirable effects. Instability in Training: GANs are known for their unstable training dynamics, which can lead to oscillations in the loss function, making convergence difficult to obtain. This issue may result in the generation of low-quality photos or training failure. GANs require a large amount of computational resources for training, especially when working with high-resolution pictures. This limitation can restrict the ability to generate high-quality synthetic faces, particularly on devices with limited computational capacity.

Generating realistic synthetic faces can create ethical and privacy problems, including the possibility for misuse in deep fakes and identity theft. These considerations must be taken into account while designing and deploying such models. Due to the absence of a comparable ground truth for synthetic faces, it can be difficult to appropriately evaluate the success of the model using typical supervised learning criteria. This shortcoming can make it difficult to evaluate the method's genuine effectiveness.

Preliminary Results

After running on the defined hyperparameters and preprocessing steps the results have come to be decent with less loss but the images are still little blur. This is because we have run the model on only 1000 images and 4 epochs. So there is a lot of noise and from the images we can see there is overfitting as well.Here the term for overfitting is defined as Mode Collapse.In this the generator gets stuck in a small subspace and learns to generate similar samples as given in the fig 1.4.

Fig 1.4: Images Generated

From fig 1.5 we can observe that the generator loss and discriminator loss is decreasing till the last epoch.

To improve its accuracy and generate more clear images we will be modifying our model architecture with more conv layers etc. More specifically converging towards State of the Art GAN models like StyleGAN,DCGAN etc.

```
Epoch : 1
/usr/local/lib/python3.9/dist-packages/keras/backend.py:5703: UserWarning:
  output, from_logits = _get_logits(
 Time:60.0
Generator Loss: 1.9472038745880127 Discriminator Loss: 0.2859085202217102

Epoch : 2
 Time:45.0
Generator Loss: 0.06550857424736023 Discriminator Loss: 4.071514129638672

Epoch : 3
 Time:47.0
Generator Loss: 2.2715296745300293 Discriminator Loss: 2.0331032276153564

Epoch : 4
 Time:46.0
Generator Loss: 1.1502877473831177 Discriminator Loss: 0.8541467189788818
```

Fig 1.5: Training

The other two models which could be used as comparison are Variational AutoEncoders and Flow based models.

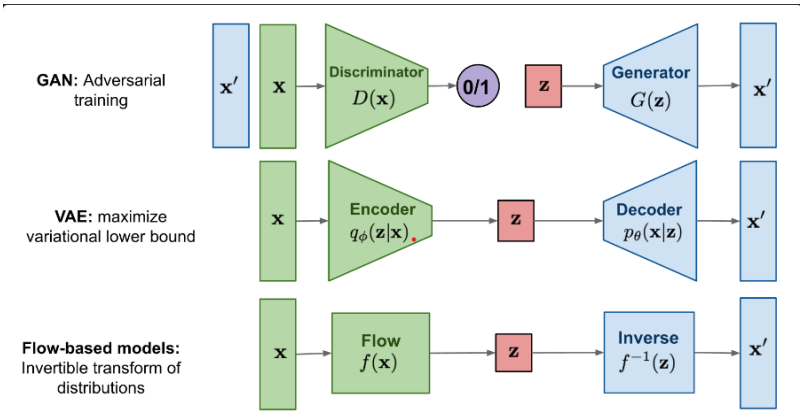

Fig 1.6 Overview of different types of generative models.

Variational AutoEncoders:

Deep neural systems called VAEs can produce fake data for datasets of images. They operate by taking a sample dataset's distribution, changing it into a new, latent space, and then returning it to the original space. An encoder-decoder architecture is what this is. The model evaluates the generated space and computes a reconstruction error, which it aims to reduce across several training cycles. VAEs use input vectors that reflect real-world images to produce output vectors that are comparable to the source images but not exact replicas of them. A VAE ensures that the output is not significantly different from the source images while introducing variability via a layer of means and standard deviations.

Flow Based Models:

Flow-based models learn how to map a more complex distribution from a simpler distribution. Typically, these models are trained to discover an invertible transformation that converts samples from a straightforward distribution to the desired distribution. The model is initially trained on a sizable collection of genuine images of faces before being used to create false images. By minimizing a suitable loss function that encourages the generated images to be indistinguishable from the real ones during training, the model develops the ability to transform samples from a basic distribution to the distribution of genuine images.