GENERATE FAKE BUT REALISTIC DATA TO INCREASE DATASET

CAPSTONE PROJECT SONY FINISHING SCHOOL

IN

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

BY

ANIL SENA
PRAVEEN RAJENDIRAN

Under the Esteemed Guidance of

Dr. Madhusudhanan

FEB - JUL 2024

SECTIONS

Abstract

Developed a Generative Adversarial Network (GAN) to generate realistic images of faces. Trained on a dataset of facial images which consists of 1024 x 1024 size enables the GAN to produce high-quality images indistinguishable from real photographs. Applications of this technology include entertainment, security, and data augmentation for various AI models. The GAN model consists of a generator and a discriminator that work together to improve the quality of the generated images. Evaluation of the model performance involves metrics to ensure the generated images meet the desired standards. Exploring further improvements and applications for the GAN-generated images remains a key focus of the project.

Acknowledgement

As we conclude our project, we extend our heartfelt thanks to God almighty for guiding us to success. Our deepest gratitude goes to everyone who supported and contributed to the completion of our project, & quot; Generating Fake but Realistic images data to increase dataset;

First, we express our sincere thanks to **SONY SOFTWARE INDIA PRIVATE LIMITED** for providing this outstanding learning opportunity and the essential resources.

We are also thankful to **IITM PRAVARTAK** for their continuous support and for creating a nurturing and conducive environment for our project.

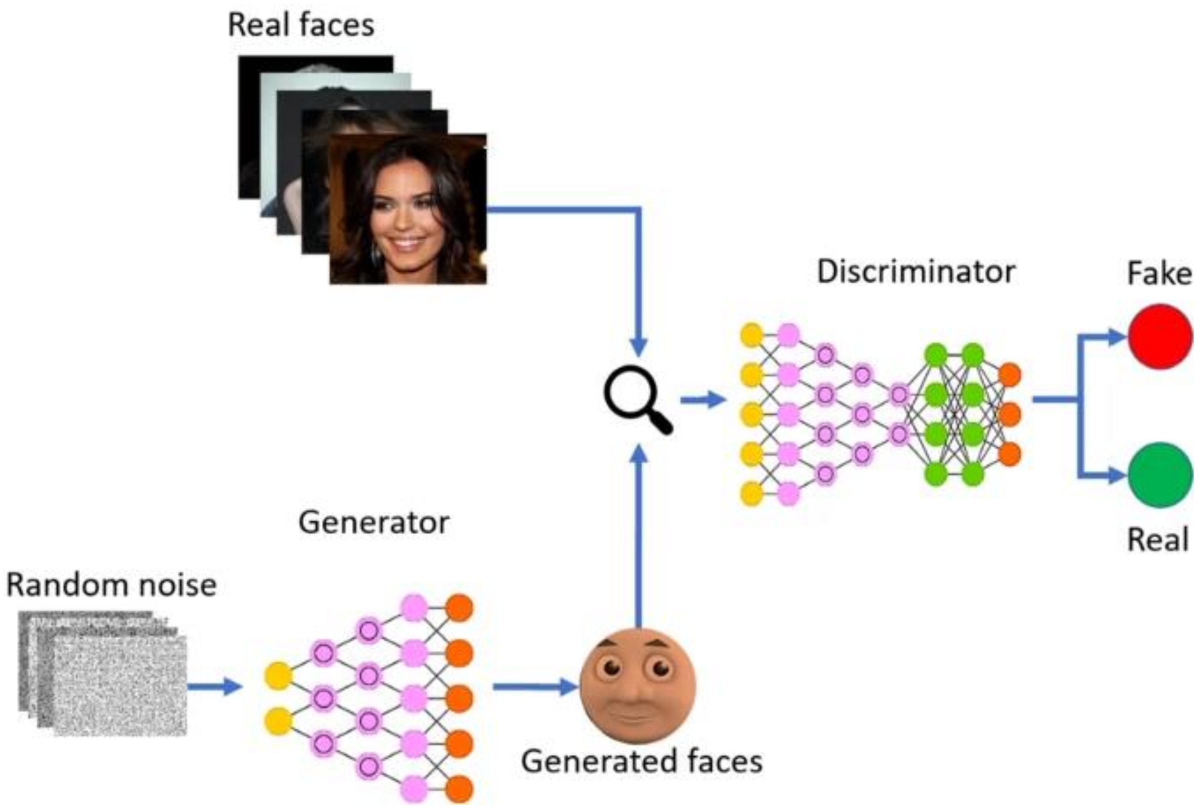
We owe a great debt of gratitude to **Dr. Madhusudhanan**, our mentor, for his invaluable guidance, constant support, and constructive feedback. His expertise and insights were critical in helping us navigate the complexities of this project and ensuring its success.

Lastly, we thank our colleagues, fellow trainees, and staff members for their collaboration, insightful discussions, and unwavering support throughout the project. Your collective efforts and encouragement were indispensable. Thank you all for making this project possible. Your collective efforts and encouragement were indispensable. Thank you all for making this project possible.

Introduction

Generative Adversarial Networks (GANs) are generative models. It uses unsupervised technique to generate new things. GAN model learns pattern in input data in such a way that they can generate new sample which resemble with the input data. The main aim of generative adversarial network is to match generated distribution with the original data distribution.

GANs are an exciting and rapidly changing field, delivering generative model 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, colouring images and in generating fake photos that even human cannot categorized as fake image.



Problem Statement

Current method for image generation often fall short in terms of realism or necessitate extensive post-processing to reach an acceptable level of quality. These limitations highlight a significant gap in the availability of efficient, end-to-end solutions that can produce high-quality, realistic images of faces without manual intervention. This project addresses this need by leveraging the power of Generative Adversarial Networks (GANs) to develop a model capable of generating lifelike facial images. By eliminating the need for post-processing, this approach streamlines the image creation process, offering a more efficient and effective solution. The success of this project demonstrates the potential of GANs to revolutionize image generation, providing a valuable tool for various applications that require realistic facial imagery.

Scope of the Project

The project involves the collection of a dataset of facial images. A GAN is trained to generate realistic images, and the quality of the generated images is evaluated. Exploration of potential applications of the generated images takes place in various fields, including entertainment, security, and data augmentation.

1. Data Pre-processing

To begin, load and pre-process dataset of images by reading the image files into an appropriate format. Resize all images to a uniform size to ensure consistency in training. Normalize the pixel values, scaling them to a range of 0 to 1, which helps in efficient training of neural networks. Finally, create TensorFlow datasets, enabling batch processing to streamline the training workflow and enhance computational efficiency.

2. Model Development

Design and implement the generator model to create synthetic images from random noise. Follow with the design and implementation of the discriminator model to evaluate the authenticity of both real and synthetic images. Define loss functions for both models to guide the training process towards better performance. Choose and configure optimizers for each model to efficiently update the model parameters during training.

3. Training Framework

Developed a training loop that updates both the generator and discriminator models iteratively, ensuring they improve together. Implement label smoothing and other techniques to stabilize training and prevent issues like mode collapse. Track and log the performance of both models during training to monitor progress and make necessary adjustments. Finally, periodically evaluate the generated images to assess the quality and improvements of the generator model.

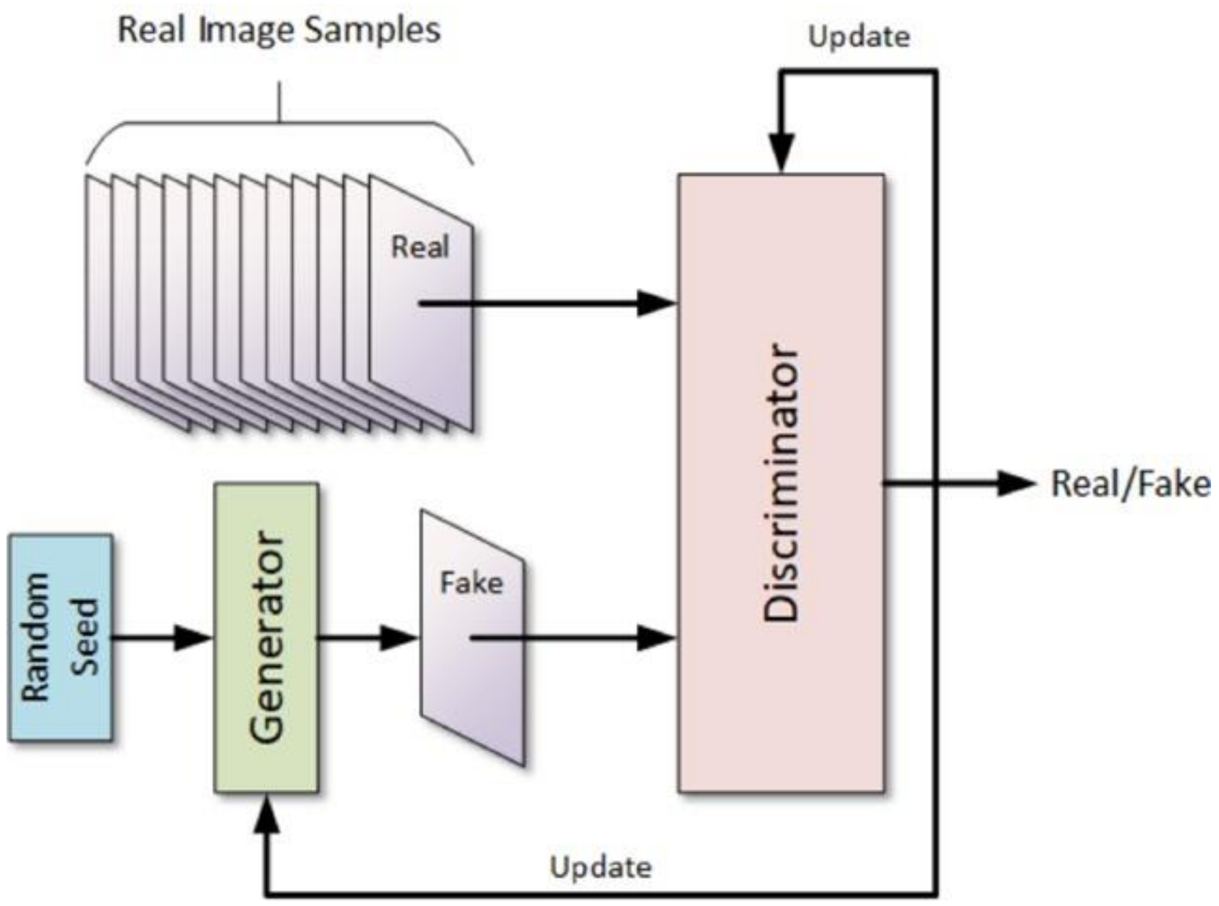
4. Evaluation

Generated images at various stages of training to evaluate the model performance and observe the progression of image quality. Plot and visualize both real and generated images to assess their realism and diversity, providing a visual benchmark. Compare the generator loss and discriminator loss over time to monitor training dynamics and ensure balanced learning between the two models. This analysis helps in identifying any issues and making necessary adjustments to the training process.

5. Optimization

Experimented with different hyperparameters, such as learning rates and batch sizes, to optimize model performance. Implement techniques like batch normalization and leaky ReLU activations to enhance the stability and efficiency of training. Analyse the impact of these changes on the quality of the generated images, assessing improvements in realism and diversity. This process helps in identifying the best configurations for producing high-quality outputs.

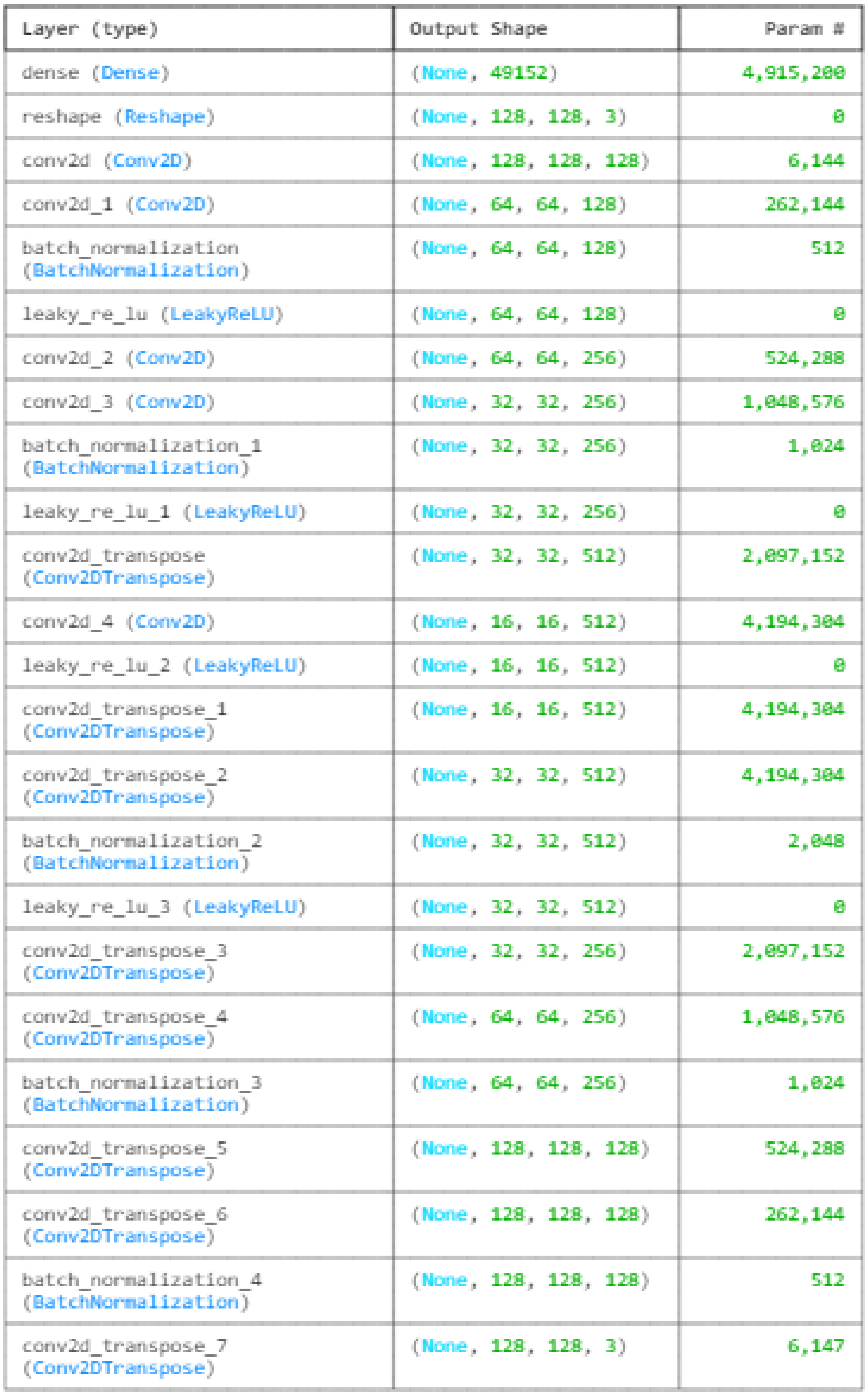
Proposed Architecture



Solutions

1. **Data Pre-processing**:
   a. Loaded images, converted them to RGB format, resized them to a fixed size (128x128 pixels), and normalized pixel values to the range [-1, 1].
   b. Converted images to arrays and created TensorFlow datasets for batch processing.

2. Generator Model:

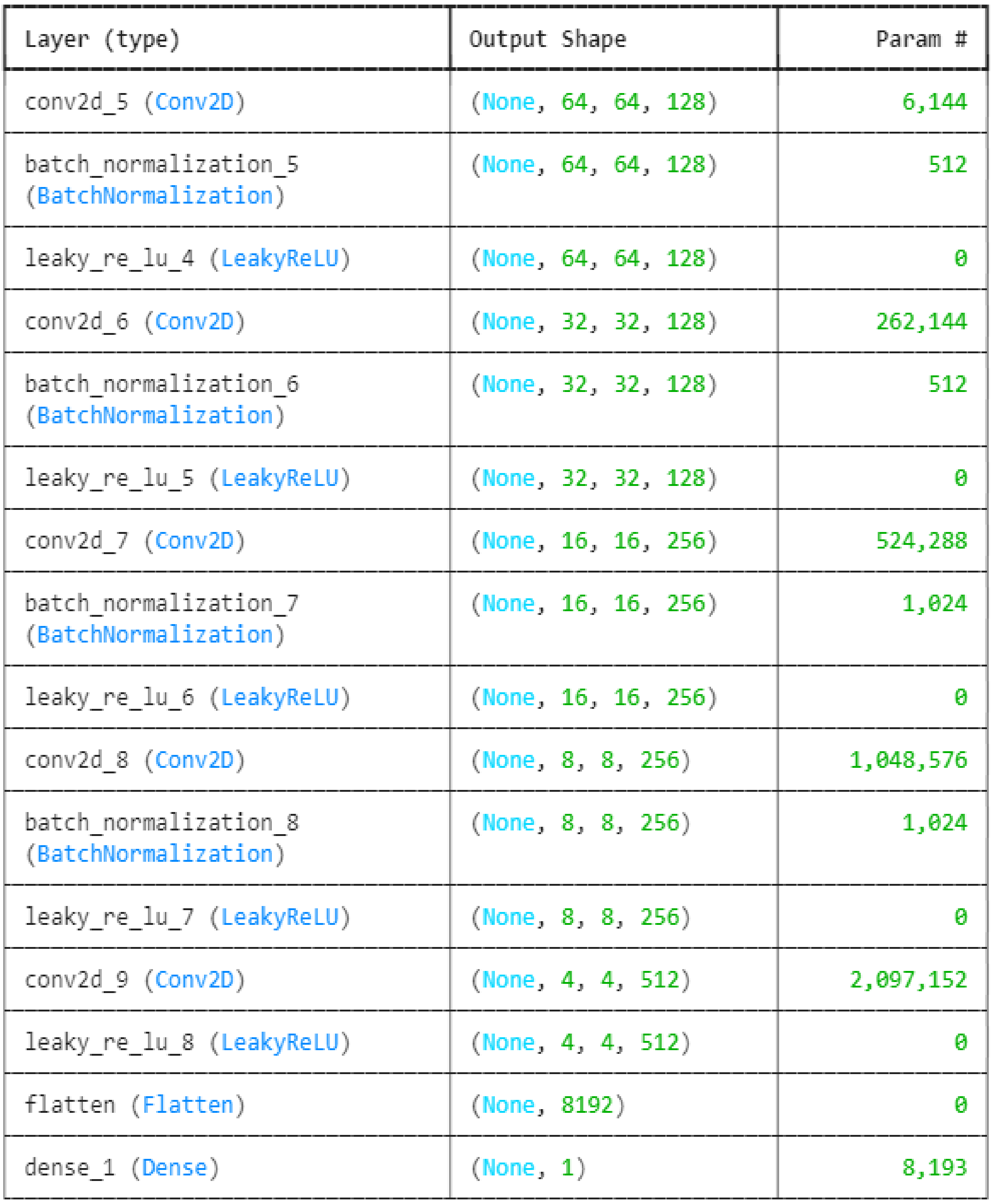


Layer (type)	Output Shape	Param #
dense (Dense)	(None, 49152)	4,915,200
reshape (Reshape)	(None, 128, 128, 3)	0
conv2d (Conv2D)	(None, 128, 128, 128)	6,144
conv2d_1 (Conv2D)	(None, 64, 64, 128)	262,144
batch_normalization (BatchNormalization)	(None, 64, 64, 128)	512
leaky_re_lu (LeakyReLU)	(None, 64, 64, 128)	0
conv2d_2 (Conv2D)	(None, 64, 64, 256)	524,288
conv2d_3 (Conv2D)	(None, 32, 32, 256)	1,048,576
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 256)	1,024
leaky_re_lu_1 (LeakyReLU)	(None, 32, 32, 256)	0
conv2d_transpose (Conv2DTranspose)	(None, 32, 32, 512)	2,097,152
conv2d_4 (Conv2D)	(None, 16, 16, 512)	4,194,304
leaky_re_lu_2 (LeakyReLU)	(None, 16, 16, 512)	0
conv2d_transpose_1 (Conv2DTranspose)	(None, 16, 16, 512)	4,194,304
conv2d_transpose_2 (Conv2DTranspose)	(None, 32, 32, 512)	4,194,304
batch_normalization_2 (BatchNormalization)	(None, 32, 32, 512)	2,048
leaky_re_lu_3 (LeakyReLU)	(None, 32, 32, 512)	0
conv2d_transpose_3 (Conv2DTranspose)	(None, 32, 32, 256)	2,097,152
conv2d_transpose_4 (Conv2DTranspose)	(None, 64, 64, 256)	1,048,576
batch_normalization_3 (BatchNormalization)	(None, 64, 64, 256)	1,024
conv2d_transpose_5 (Conv2DTranspose)	(None, 128, 128, 128)	524,288
conv2d_transpose_6 (Conv2DTranspose)	(None, 128, 128, 128)	262,144
batch_normalization_4 (BatchNormalization)	(None, 128, 128, 128)	512
conv2d_transpose_7 (Conv2DTranspose)	(None, 128, 128, 3)	6,147

This image shows a summary of a neural network model

a. Designed a generator model using dense layers followed by reshaping and several convolutional and up sampling layers.
b. Incorporated batch normalization and leaky ReLU activations to enhance model performance.
c. Used a tanh activation function in the final layer to produce pixel values in the range [-1, 1].

3. Discriminator Model:



```
┃ Layer (type)                      ┃ Output Shape            ┃        Param # ┃
│ conv2d_5 (Conv2D)                 │ (None, 64, 64, 128)     │          6,144 │
│ batch_normalization_5             │ (None, 64, 64, 128)     │            512 │
│ (BatchNormalization)              │                         │                │
│ leaky_re_lu_4 (LeakyReLU)         │ (None, 64, 64, 128)     │              0 │
│ conv2d_6 (Conv2D)                 │ (None, 32, 32, 128)     │        262,144 │
│ batch_normalization_6             │ (None, 32, 32, 128)     │            512 │
│ (BatchNormalization)              │                         │                │
│ leaky_re_lu_5 (LeakyReLU)         │ (None, 32, 32, 128)     │              0 │
│ conv2d_7 (Conv2D)                 │ (None, 16, 16, 256)     │        524,288 │
│ batch_normalization_7             │ (None, 16, 16, 256)     │          1,024 │
│ (BatchNormalization)              │                         │                │
│ leaky_re_lu_6 (LeakyReLU)         │ (None, 16, 16, 256)     │              0 │
│ conv2d_8 (Conv2D)                 │ (None, 8, 8, 256)       │      1,048,576 │
│ batch_normalization_8             │ (None, 8, 8, 256)       │          1,024 │
│ (BatchNormalization)              │                         │                │
│ leaky_re_lu_7 (LeakyReLU)         │ (None, 8, 8, 256)       │              0 │
│ conv2d_9 (Conv2D)                 │ (None, 4, 4, 512)       │      2,097,152 │
│ leaky_re_lu_8 (LeakyReLU)         │ (None, 4, 4, 512)       │              0 │
│ flatten (Flatten)                 │ (None, 8192)            │              0 │
│ dense_1 (Dense)                   │ (None, 1)               │          8,193 │
```

This image shows a summary of a neural network model

a. Constructed a discriminator model using convolutional layers to extract features from images.
b. Applied batch normalization and leaky ReLU activations for stability and performance.

   c. Used a sigmoid activation function in the final layer to output a probability indicating whether an image is real or fake.

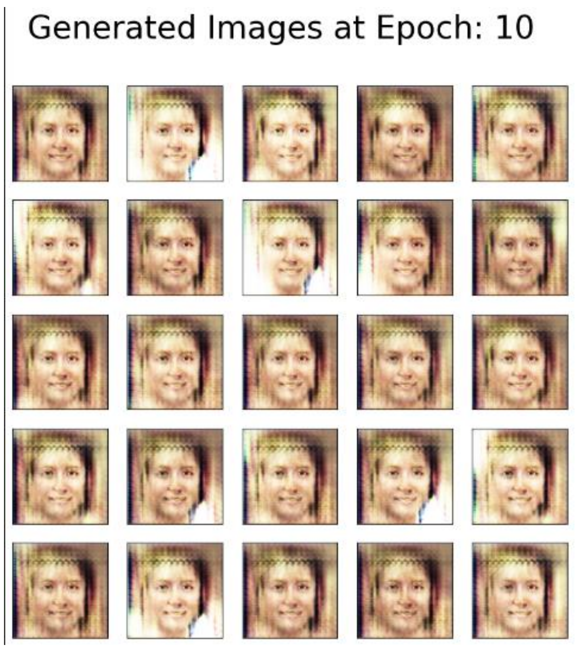
4. **Training Loop**:
   a. Developed a training loop that alternates between updating the generator and discriminator.
   b. Used label smoothing in the loss functions to prevent the discriminator from becoming too confident.
   c. Computed gradients using TensorFlow's Gradient Tape and applied them using Adam optimizers.
5. **Evaluation and Visualization**:
   a. Implemented functions to plot real images and generated images for visual assessment.
   b. Generated images at the end of each epoch to track the progress of the generator.
   c. Displayed generator and discriminator losses to monitor training dynamics and adjust hyperparameters if necessary.

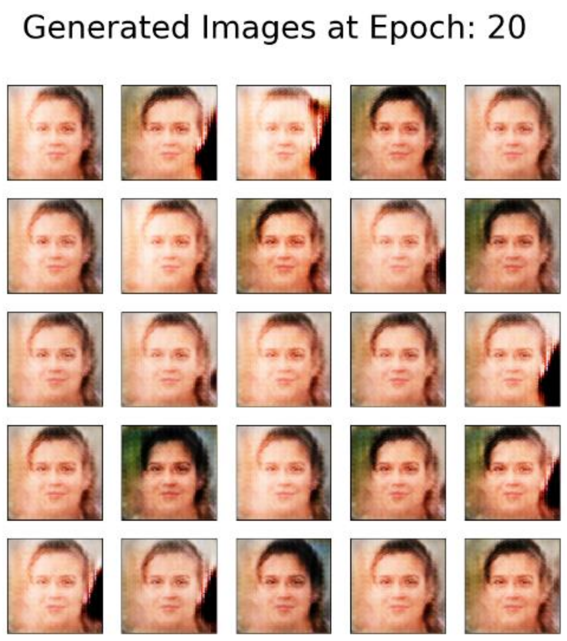
Results

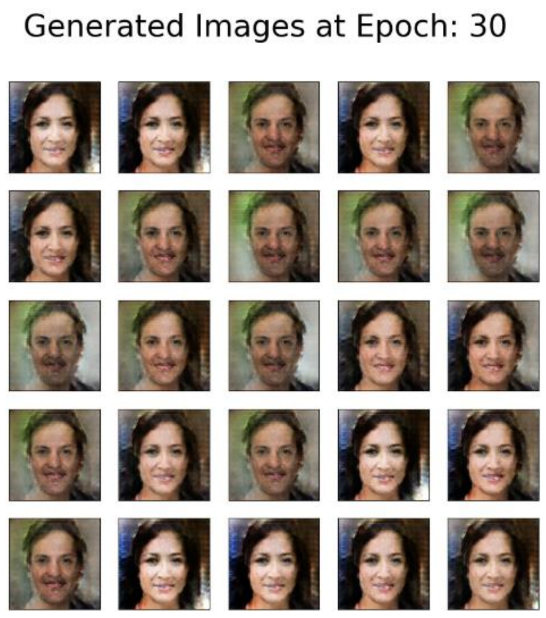
Main Observations

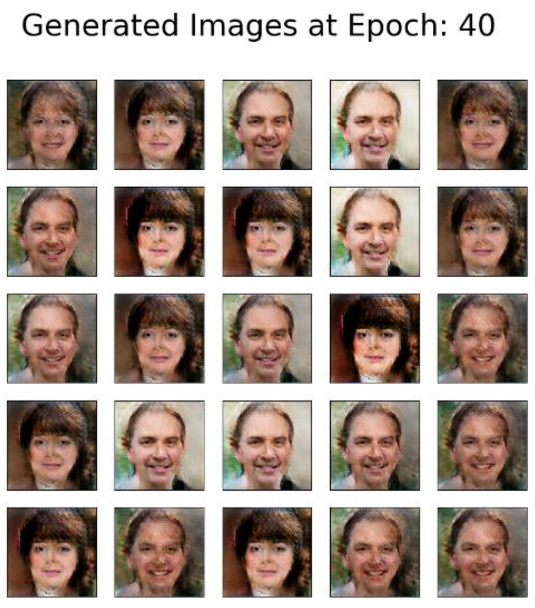
The GAN model successfully generated realistic images of faces. The quality of the generated images improved significantly with increased training epochs.

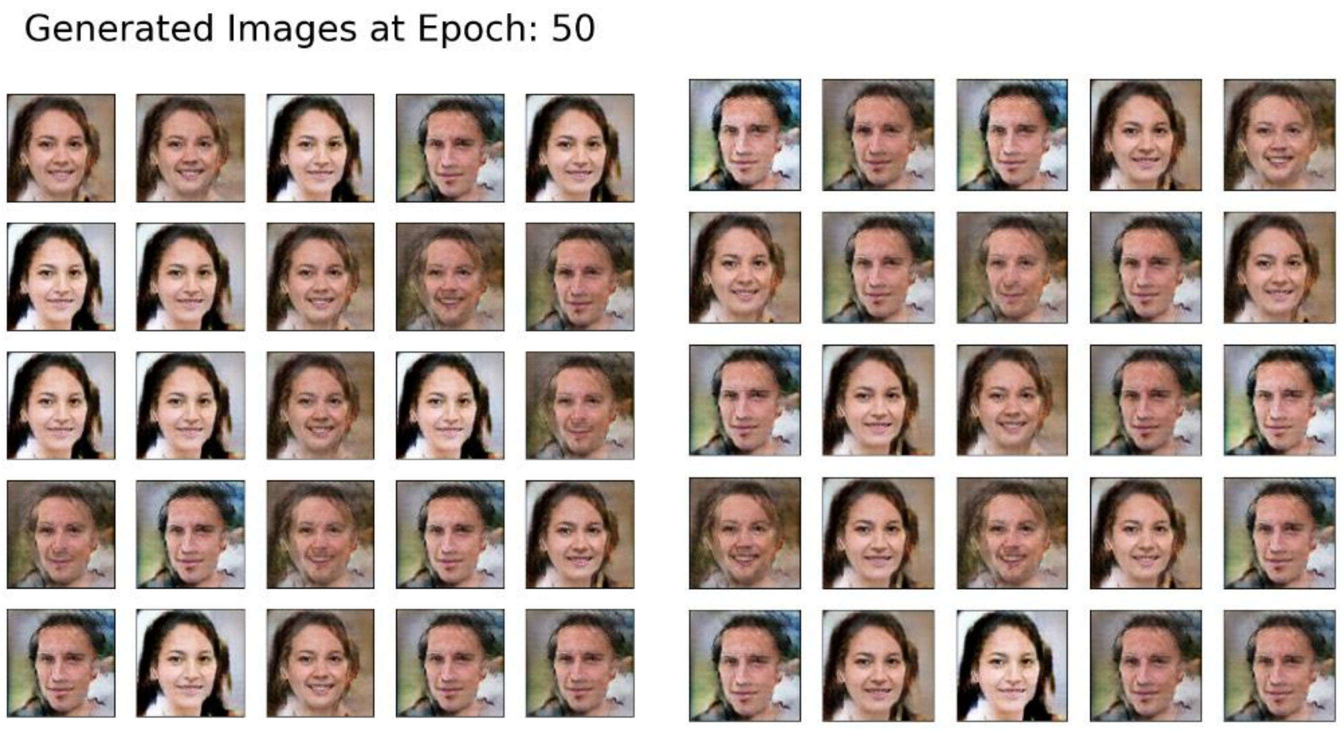
Visualization of Results











Features

1. **Alphanumeric Sorting**:
    - Ensures that image files are loaded in a consistent order, which is crucial for reproducibility and debugging.
2. **Image Pre-processing**:
    - Resizes images to 128x128 pixels, converts them to RGB, and normalizes pixel values to [-1, 1].
    - Converts images to arrays and batches them using TensorFlow datasets.
3. **Model Summaries**:
    - Provides detailed summaries of the generator and discriminator architectures, including the shape and number of parameters in each layer.

4. **Optimized Training**:
    - Utilizes the Adam optimizer with a learning rate of 0.0001 and beta values of 0.5 and 0.999 for effective training.
    - Applies label smoothing in the loss functions to enhance training stability.
5. **Loss Functions**:
    - Uses binary cross-entropy loss with label smoothing for both the generator and discriminator to stabilize training.
6. **Training Visualization**:
    - Plots real images and generated images at different training stages to visually assess the progress of the GAN.
7. **Batch Processing**:
    - Uses TensorFlow datasets for efficient batch processing during training, which helps in managing memory and speeding up training.

Conclusion

The project demonstrates the effectiveness of Generative Adversarial Networks (GANs) in generating highly realistic images of faces. The results show that GANs can produce images with fine details and natural features, making them almost indistinguishable from real photographs. This capability opens up a multitude of applications, including virtual reality, where lifelike avatars can enhance user experiences, and digital art, providing artists with new creative tools. Additionally, GANs can significantly improve training datasets for AI models by generating a diverse array of images, addressing data scarcity issues and improving model performance. The promising results from this project highlight the potential of GANs in revolutionizing fields that rely on realistic image generation, pushing the boundaries of what is possible in computer vision and artificial intelligence. The continuous advancement in GAN technology is set to drive innovation and bring about transformative changes across various industries.

Future Scope

Future work could involve improving the realism of the generated images further, exploring other GAN architectures, and expanding the dataset to include more diverse facial features. Additionally, integrating the GAN model with real-time applications could open new avenues for interactive and immersive experiences.