DATS Machine Learning II
Dr Amir Jafari

Face Image Generation

Mohammad Maaz

Introduction

Our final project involves using generative deep learning to generate human facial images. I selected this problem because synthetic generation of image data is a highly sensitive topic in the fields of surveillance and intelligence; according to the Associated Press, U.S. lawmakers first met to discuss the issue of artificially generated imagery in 2019. The prevalence of synthetic image data has only increased since. I implement three model architectures as follows.

1. Variational Autoencoder
2. Generative Adversarial Network
3. Wasserstein Generative Adversarial Network

I will compare the results of the three models and select the best generated human facial images.

Individual Work

My responsibility in the project was the implementation of Generative adversarial networks (GANs). These are a family of deep learning algorithms that are designed to generate new data that is similar to a training dataset.

Architecture

The architecture of GANs typically consists of two neural networks: a generator and a discriminator. The generator network is trained to generate new data that is similar to the training data, while the discriminator network is trained to distinguish real data from fake data. The two networks are trained together in a zero-sum game, where the generator tries to produce data that the discriminator cannot distinguish from the real data, and the discriminator tries to correctly identify the fake data produced by the generator. The pseudocode for this algorithm is as follows.

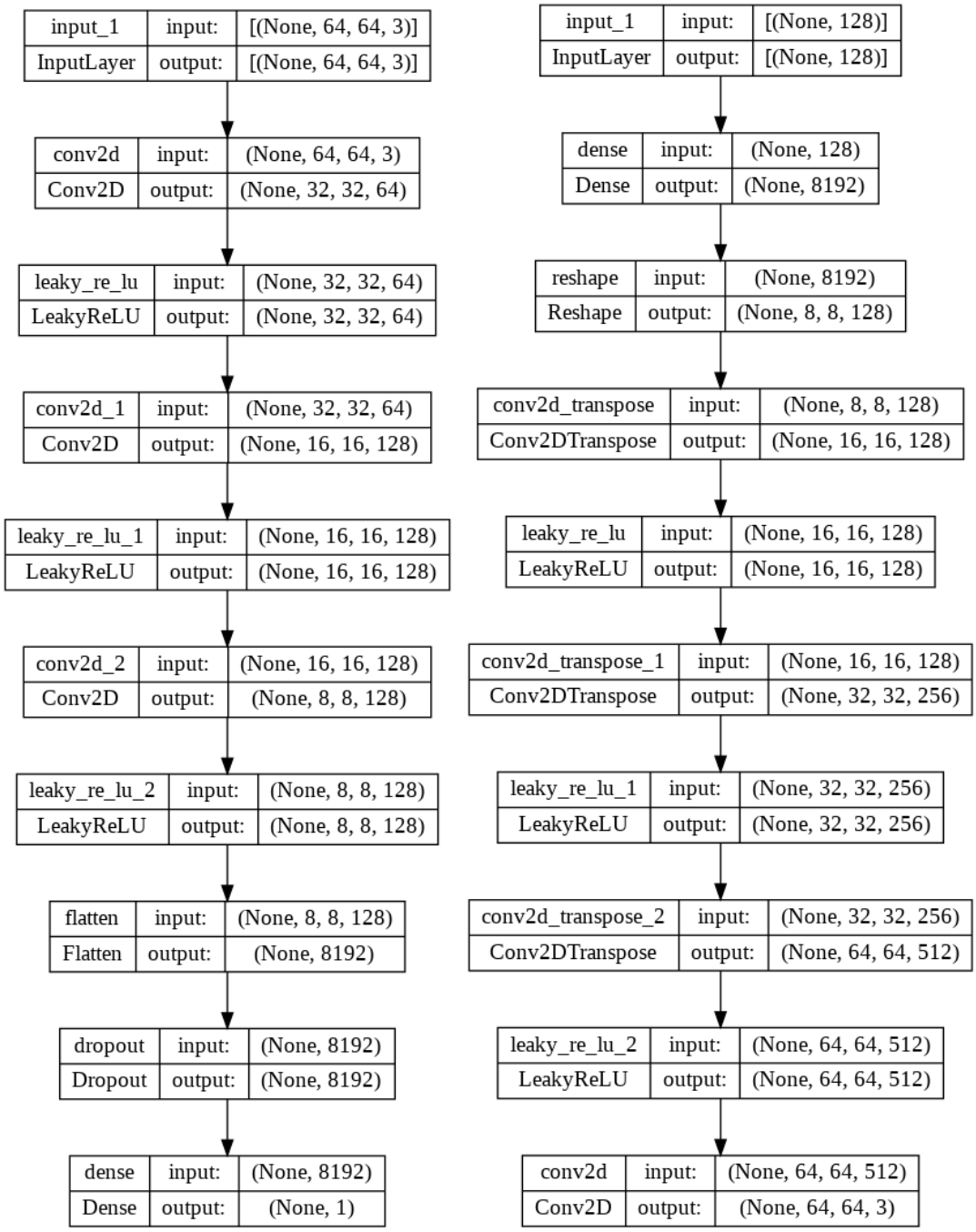
1. Initialize the generator and discriminator networks
2. For each batch of real images:
    a. Generate a batch of fake images using the generator
    b. Train the discriminator on the fake and real data

c. Train the generator on a batch of fake data, using the output of the discriminator as feedback

3. Repeat this process until the generator produces data that is indistinguishable from the real data

Network

The discriminator (left) and generator (right) networks are defined as follows.



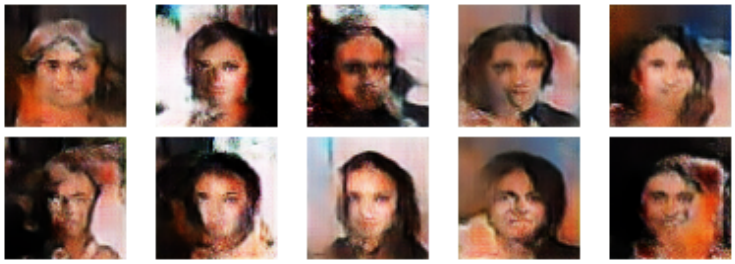
Experiment

I train the model on the CelebA dataset. For preprocessing, I resize the images to 64x64 pixels and divide them by 255 to get floating points for pixel values. I then run the model on 30 epochs. One epoch takes approximately 40 minutes to complete, so the experiment took around 20 hours. I implemented a custom callback that tests the generator after every epoch, allowing us to visualize the generated images and keep track of discriminator and generator losses.

Epoch: 10 Disc loss: 0.65 Gen loss: 1.06

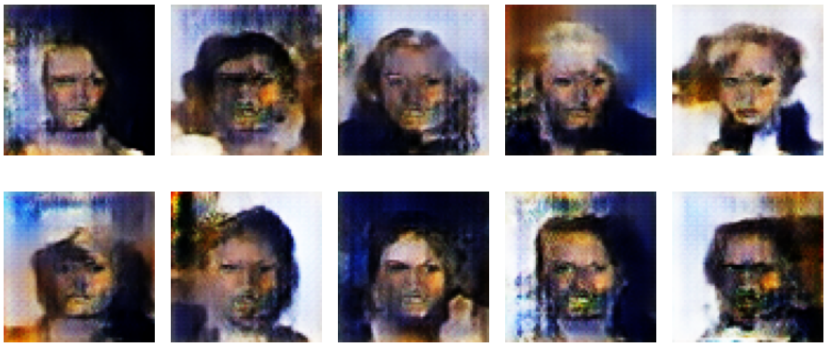
Epoch: 20 Disc loss: 0.60 Gen loss: 0.99

Epoch: 30 Disc loss: 0.55 Gen loss: 0.85



Results

The custom callback also saves the generator network periodically, allowing us to run inferences when I want. The results for the generator after 50 epochs are as follows.



As I can see, the images are distorted versions of human faces. It is quite easy to tell real and generated images apart.

Percentage of Code Copied

$= \frac{103-59}{103+63} \times 100 = 26.5\%$

Summary and Conclusion

I can see that our network was successful in generating images resembling faces, but the quality was lacking. This can be due to several reasons.

- Inadequate training: GANs take a long time to train, but the results keep improving after each epoch. In our experiment more epochs might have produced better results.
- Training instability: GANs can be difficult to train, and even small changes in the training process can affect the outcome. In our experiment, we were not able to experiment with hyperparameters, which might have led to non-optimal hyperparameters being used.

References

Cheong, Soon Yau. *Hands-On Image Generation with TensorFlow: A Practical Guide to Generating Images and Videos Using Deep Learning*. Packt Publishing, 2020.

Chollet, Francois. “DCGAN to generate face images.” *Keras*, 29 April 2019, https://keras.io/examples/generative/dcgan_overriding_train_step/. Accessed 12 December 2022.