Generating anime faces using GAN's

Fuzel Ahamed Shaik
University of Oulu
Oulu, Finland
fshaik22@student.oulu.fi

Abstract

Generative Adversarial Networks shortly know as GAN's are newly developed architecture to automatically discover or capture the patterns from the input data. It is an unsupervised learning algorithm which leverages convolutional neural network architecture to produce the results. Here in this paper, we are going to utilize GANs to automatically create anime character faces from a given input dataset with the anime character faces. This modelling creates a new approach termed as generative modelling.

1. Introduction

In recent times, GANs have become a go to approach to generate artificial data majorly in the computer vision domain. For example, one can use GAN architecture to create auto generative characters which fits the description of the given input [1]. This gave the idea to leverage this popular model to generate artificial anime character faces. As the popularity of anime is increase now-a-days, the effort to draw and create the characters in the design software is going to be a very hectic task. With help of this architecture one can easily create anime character faces with good detail. This can reduce the huge task and reduce the time for the process.

2. Methodology

The applications of GANs such as generating synthetic training data, creating arts, style-transfer, image-to-image translation, etc., intrigued machine learning community to explore on this model. The architecture consists of 2 networks namely the generator and the discriminator. The generator tries to generate fake samples and make discriminator believe the samples to be real [2]. The discriminator then tries to detect the generated samples from both the real and fake samples [1]. To simplify, the generator generates the synthetic data by inserting some random noise while the discriminator acts as a binary classifier and classifies the input sample as real or fake.

2.1. Dataset

The dataset is downloaded from Kaggle's public image dataset repository [4]. The link to the dataset is added in the references. The downloaded dataset consists of 36740 high-quality images of anime faces. All the images are colored images with the size of 64 x 64 pixels.

Figure 1. Sample images from the dataset.

2.2. Preprocessing and loading the data

Data preprocessing is one of the main tasks before feeding the input to the model. The preprocessing methods involve in this project are to resize the image into 64 with bilinear interpolation, later transforms the image to a tensor object and finally transform the images into values where the mean of image data is 0.5 and standard deviation of the image is 0.5. Define a class named AnimeData to extract the dataset and load the training images using DataLoader.

2.3. Defining GAN architecture

GAN architecture consists of two CNNs to generate images namely, generator and discriminator. In this project, the generator is constructed with 5 transpose convolutional layers which helps in up-sampling the image during training process. Batch normalization techniques is

used to standardize the inputs to the layers [3]. ReLU is used as activation function for hidden layers whereas Tanh is used as activation function for the output layer. The output layer is the new image as the same size as in the training data samples i.e., 3x64x64.

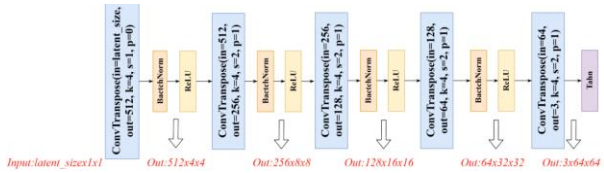

Figure 2. Generator architecture

The discriminator is also constructed with similar architecture as the generator by using batch normalization, LeakyReLU activation function for hidden layers, and sigmoid activation function for output layer. The inputs of the discriminator are 3x64x64 tensor images, and the discriminator gives single values as an output which indicates whether the given image is fake or not.

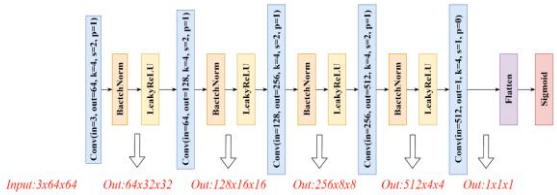

Figure 3. Discriminator architecture

2.4. Training the model

The neural network is trained with preprocessed images for 40 iterations. The training process mainly involves training discriminator, training generator, and saving the model. Discriminator loss is the sum of real loss and fake loss. The real loss is calculated by feeding the real images to the discriminator whereas, fake loss is the loss generated feeding the generated fake image by inserting noise to the generator. The generator tries to fool the discriminator by using the real loss function which is the same real loss function that used to train the discriminator.

3. Results and discussion

```
Epoch [40/40], loss_g: 3.6386, loss_d: 0.5320, real_score: 0.9120, fake_score: 0.1227
```

Figure 4. Final result of real and fake loss after 40 epochs

The training loss of the discriminator stays low whereas generator loss is quite stable. The losses of generator and discriminator are too far from each other to provide optimal training and performance of the network.

Figure 5. Resultant images

4. Conclusion

To conclude, the samples created got better eventually, but couldn't notice any improvements in the latter epochs. A deeper architecture would have benefited from greater number of epochs as it would be able to optimize more parameters. For further modeling one can try using different optimizers to generator and discriminator. To improve the performance of this model one can try to increase the dropout ratios in the discriminator architecture. This can solve the vanishing gradient problem. More hyperparameter techniques such as smaller learning rates, adding noise to the discriminator layers.

References

[1] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta and A. A. Bharath, "Generative Adversarial Networks: An Overview," in *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53-65, Jan. 2018, doi: 10.1109/MSP.2017.2765202.

[2] J. Gui, Z. Sun, Y. Wen, D. Tao and J. Ye, "A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications," in *IEEE Transactions on Knowledge and Data Engineering*, doi: 10.1109/TKDE.2021.3130191.

[3] G. Antipov, M. Baccouche and J. -L. Dugelay, "Face aging with conditional generative adversarial networks," *2017 IEEE International Conference on Image Processing (ICIP)*, Beijing, China, 2017, pp. 2089-2093, doi: 10.1109/ICIP.2017.8296650.

[4] Kaggle. 2021. Retrieved from https://www.kaggle.com