Anime Face Generation

Mahdi Akbari (#2208094)
University of Oulu, Finland
mahdi.akbari@.oulu.fi

Abstract

Generative Adversarial Network (GAN) is an unsupervised machine learning model using deep learning. In this project, GAN is used to extract features from Anime Face Dataset to train Generative (G) and discriminative (D) networks. In this project, both networks have five transpose convolutional layers. D network helps G to capture the samples distribution and G maximizes the probability of mistake in D. Thus, D tries to minimize mistakes ending to minmax optimization. Both networks contest with each other in a zero-sum game. Pictures developed by G are mimicking inputs having some noise. At the end, D learns to detect fake images too. We found that increasing epochs improve quality of generated images by G.

1. Introduction

Generative Adversarial Networks (GANs) are an emerging technique for both semi supervised and unsupervised learning. They achieve this through implicitly modeling high-dimensional distributions of data. Proposed in 2014 [1], they can be characterized by training a pair of networks in competition with each other. A common analogy, apt for visual data, is to think of one network as an art forger and the other as an art expert. The forger, known in the GAN literature as the generator, G, creates forgeries, with the aim of making realistic images. The expert, known as the discriminator, D, receives both forgeries and real (authentic) images, and aims to tell them apart (Figure 1). Both are trained simultaneously, and in competition with each other. Crucially, the generator has no direct access to real images— the only way it learns is through its interaction with the discriminator. The discriminator has access to both the synthetic samples and samples drawn from the stack of real images. The error signal to the discriminator is provided through the simple ground truth of knowing whether the image came from the real stack or from the generator. The same error signal, via the discriminator, can be used to train the generator, leading it toward being able to produce forgeries of better quality [2].

Generative models learn to capture the statistical distribution of training data, allowing us to synthesize samples from the learned distribution. On top of synthesizing novel data samples, which may be used for downstream tasks such as semantic image editing [3], data augmentation [4], and style transfer [5], we are also interested in using the representations that such models learn for tasks such as classification [6] and image retrieval [7]. We occasionally refer to fully connected and convolutional layers of deep networks; these are generalizations of perceptrons or spatial filter banks with nonlinear postprocessing. In all cases, the network weights are learned through backpropagation [7]. In this project, GAN using deep convolutional layers is utilized to generate fake images from anime faces.

2. Data

Anime Face Dataset NTU-MLDS was used from Kaggle Datasets (https://www.kaggle.com/datasets). The dataset consists of 36.7k cartoon (Figure 2). Before any modeling, preprocessing was performed: cropping at the center, resize into 64*64 array with bilinear interpolation, converts (a pixel range [0, 255]) to a tensor, and transform to mean and standard deviation equal to 0.5 and normalize images in [-1, 1].

3. Methods

3.1. Discriminator Network

The discriminator network (D in Figure 1b) is characterized as a function that maps from image data to a probability that the image is from the real data distribution, rather than the generator distribution. For a fixed generator, G, the discriminator, D, may be trained to classify images as either being from the training data (real, close to one) or from a fixed generator (fake, close to zero). When the discriminator is optimal, it may be frozen, and the generator, G, may continue to be trained so as to lower the accuracy of the discriminator. If the generator distribution is able to match the real data distribution perfectly, then the discriminator will be maximally confused, predicting 0.5 for all inputs. In practice, the discriminator might not be trained until it is optimal; we explore the training process in more depth in the section “Training GANs.”

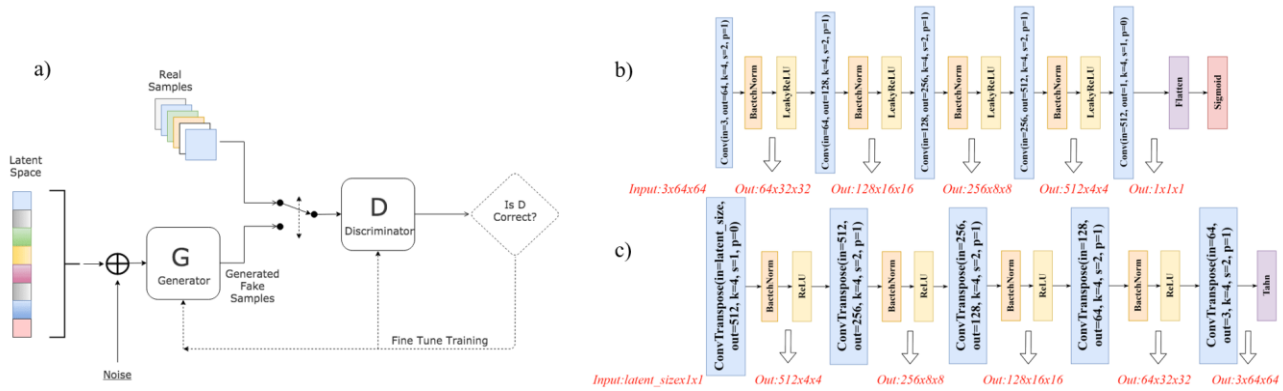

Figure 1: GAN: Generative Adversarial Networks (a), D: Discriminator (b), and G: Generator (c)

Figure 2: Sample input images from Anime Face Dataset

3.2. Generator Network

In this project, 2D transposed convolution followed by a batch normalization and RelU activation function was utilized. G generate a new RGB image of the same size as our training data (3x64x64). Five transpose convolutional layers for both D (Figure 1b) and G(Figure 1a) were defined. The batch size was 128.

3.3. Loss function and optimizer

D and G networks are trained at the same time. The loss function of them is mean squared error (MSE). Discriminator's loss is the sum of loss for real and fake images. For the initial experiment, Adam optimizer for both networks were defined. Also, ten epochs were introduced (images saved in generated folder), because more epochs were too expensive for my laptop to be run. The quality of output in the final epoch was good enough (Figure 4).

3.4. Training the discriminator

The goal to train the discriminator is to maximize the probability of correct classification. Practically we are going to maximize log(D(x)) + log(1-D(G(x))) which we call it the objective function for the first part of training. We will calculate this objective function in two steps. First, we create a batch of real samples from the training set., pass it through forward pass (D). Then we calculate the first part of objective function. In the next step, we will calculate the gradients in the backward pass. The next step is to calculate the second term of the objective function. To this end, we will construct a batch of fake samples using the generator network. Then pass it through the forward pass, then calculate the second term. Now, we are ready to accumulate these two terms and calculate the objective function. After this step we can call a step of the optimizer of the discriminator [1].

3.5. Training the Generator

The goal to train the generator is to minimize log(1-D(G(x)) to have a better fake image. Minimizing log(1-D(G(x)) is equal to maximizing log(D(G(x)). Therefore, this is our objective function here. To this end, we will pass the constructed fake batch in previous step through the discriminator. Then calculate the loss and compute the gradients of backward pass. Finally, we call a step of the generator's optimizer. Using the real labels here to calculate the objective function's loss directly translated to the calculating the loss for the term log(1-D(G(x)) which is the term that we are looking for [1].

4. Results and discussion

As shown in Figure 3, both generator and discriminator are trying to have the loss around 0.5. Loss of G and D are 0.51 and 0.05 respectively in the last epoch. Also, real and fake scores are 0.95 and 0.2 respectively in epoch number 10. Based on the last epoch results (Figure 4), G network was successful to increase the quality of fake images by increasing epochs. In the first epoch, fake images were very noisy. In epoch 10, the G produced fake images with more details and higher diversity. Use a dataset with more diversity will end to better generated fake images. Using more diversified database (e.g., black faces cartoon) will lead to finding more feature.

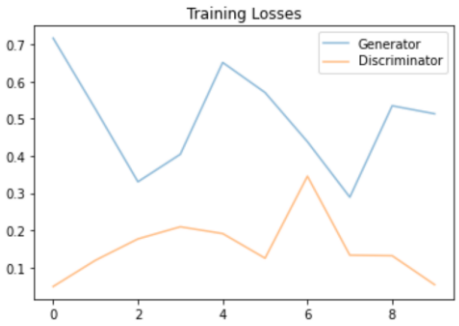

Figure 3: The Genrator and Discreminator networks losses

Figure 4: Fake images generated in the last (10th) epoch

5. Conclusion

Using Anime Face Dataset NTU-MLDS (kaggle website having 36.7k high-quality images), Generative Adversarial Networks (GANs) was used in this project to produce fake cartoon faces. GAN is a generative unsupervised learning model which trains generative (G) and discriminative (D) networks. Based on literature, GAN has proven advantageous for generating fake images. In this project, by 10 epochs in training G and D, good quality fake images were produced (having some level of noise).

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Proc. Advances Neural Information Processing Systems Conf., 2014, pp. 2672–2680.

[2] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview. IEEE signal processing magazine, 35(1), 53-65.

[3] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in Proc. European Conf. Computer Vision, 2016, pp. 597–613.

[4] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in Proc. IEEE Conf. Computer Vision Pattern Recognition, 2016, pp. 3722–3731.

[5] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks, Proc. Int. Conf. Computer Vision. [Online]. Available: https://arxiv.org/abs/1703.10593

[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in Proc. 5th Int. Conf. Learning Representations Workshop Track, 2016.

[7] A. Creswell and A. A. Bharath, "Adversarial training for sketch retrieval," in Proc. European Conf. Computer Vision Workshops, Amsterdam, The Netherlands, 2016, pp. 798–809.