An Implementation of GAN and VAE

Yubo Wang
CISE Department
University of Florida
yubo.wang@ufl.edu

Abstract

With the continuous development of computer vision technology, image generation technology has gradually become an important research paper in computer vision. This paper implements GAN (Generative Adversarial Network) and VAE (Variational Auto-Encoder) algorithm respectively. Experiments are carried out on the MNIST handwritten digital data set, and the results are compared. (Related code could be found at: https://github.com/Wyb0627/ML_project_GAN_VAE)

1 Introduction

With the advent of big data and intelligence, image acquisition tools in various fields have provided us with a huge amount of image data and important help for in-depth understanding and research in a professional field. Image data still have problems in the specific research field, such as lack of acquisition technology, high acquisition cost, and acquisition data defects. Therefore, establishing an effective natural image generation model has become one of the critical issues of computer vision. Its goal is to change some potential parameters according to natural images' distribution to generate different real images. Therefore, there exists the need to construct a desired generative model to capture the underlying data distribution. There are two main categories of image generation models with great potential today, namely GAN (Generative Adversarial Networks) and VAE (Variational Autoencoders).

2 Implementations

This section would carry out the model structure, and a brief introduction of the GAN and VAE implemented.

2.1 Implementation of GAN

GAN was proposed by Goodfellow et al. in 2014. This network has strong generation capabilities and high generation quality and has attracted great attention in generative model research. GAN trains the generative model through the confrontation process. The two networks participating in the confrontation training are Generator and Discriminator[1]: Generator is a network that generates images. It receives a random noise z and generates images through this noise, denoted as $G(z)$. Discriminator is A network for judging an image, used to judge an image's authenticity. Its input parameter is x, x represents an image, and the output $D(x)$ represents the probability that x is a real image. If it is 1, it represents 100% is a real image, and the output is 0, which means that it cannot be a real image.

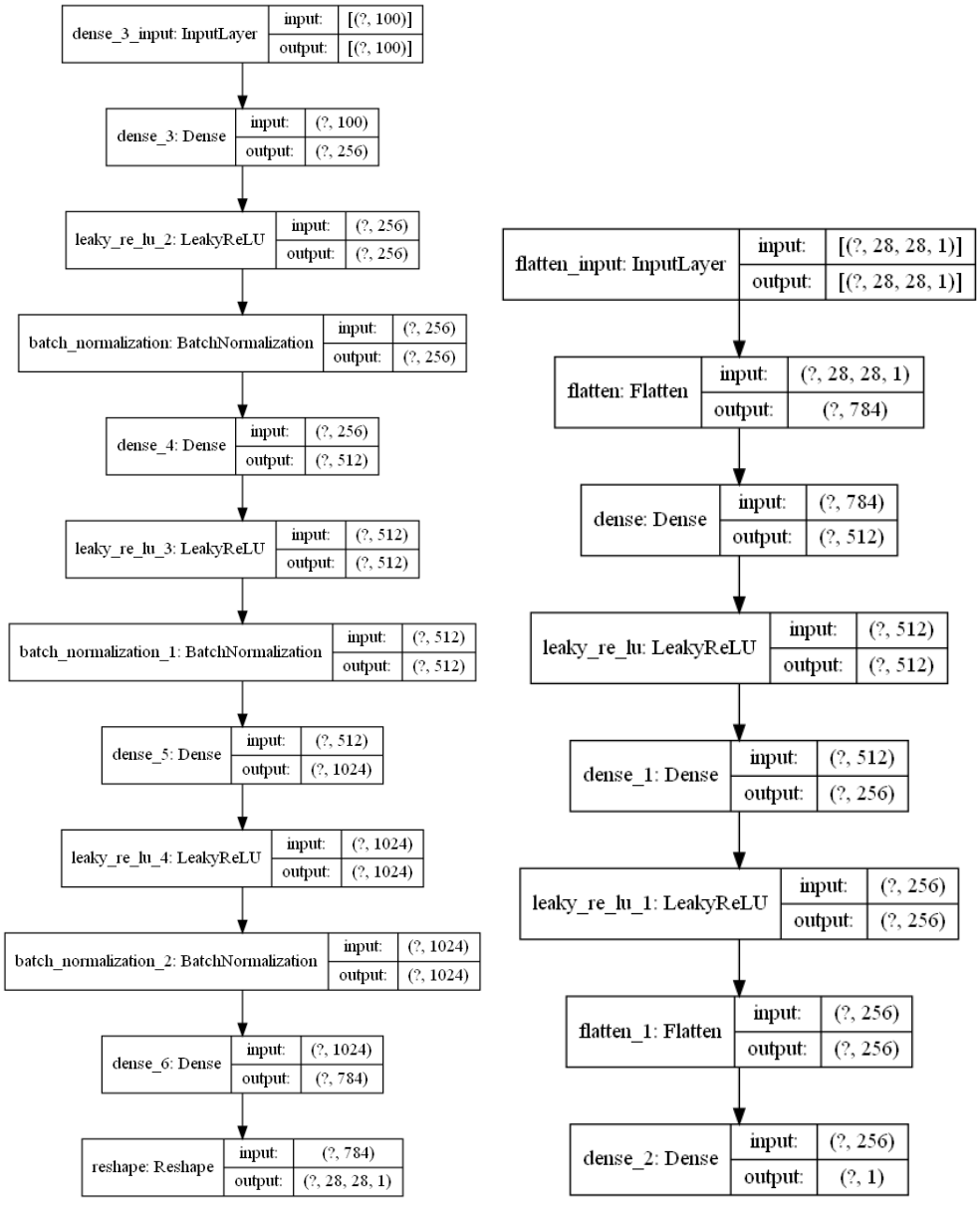
In this paper, a DCGAN was implemented. In DCGAN, both the discriminator and generator use Convolutional Neural Network (CNN) to replace the multi-layer perceptron in GAN. Simultaneously, to make the entire network differentiable, the pooling layer in CNN is removed. The connection layer is replaced with a global pooling layer to reduce the amount of calculation.

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

In terms of training, the training method of DCGAN is the same as GAN. By training D to enhance the discriminative ability of the discriminator, maximize the probability of assigning the correct label, that is, maximize $log(D(x))$, and train G or $log(1 - D(G(z)))$ to minimize. The training of discriminator and generator can be expressed as a Nash Equilibrium problem between them, its expression is as formula (1).

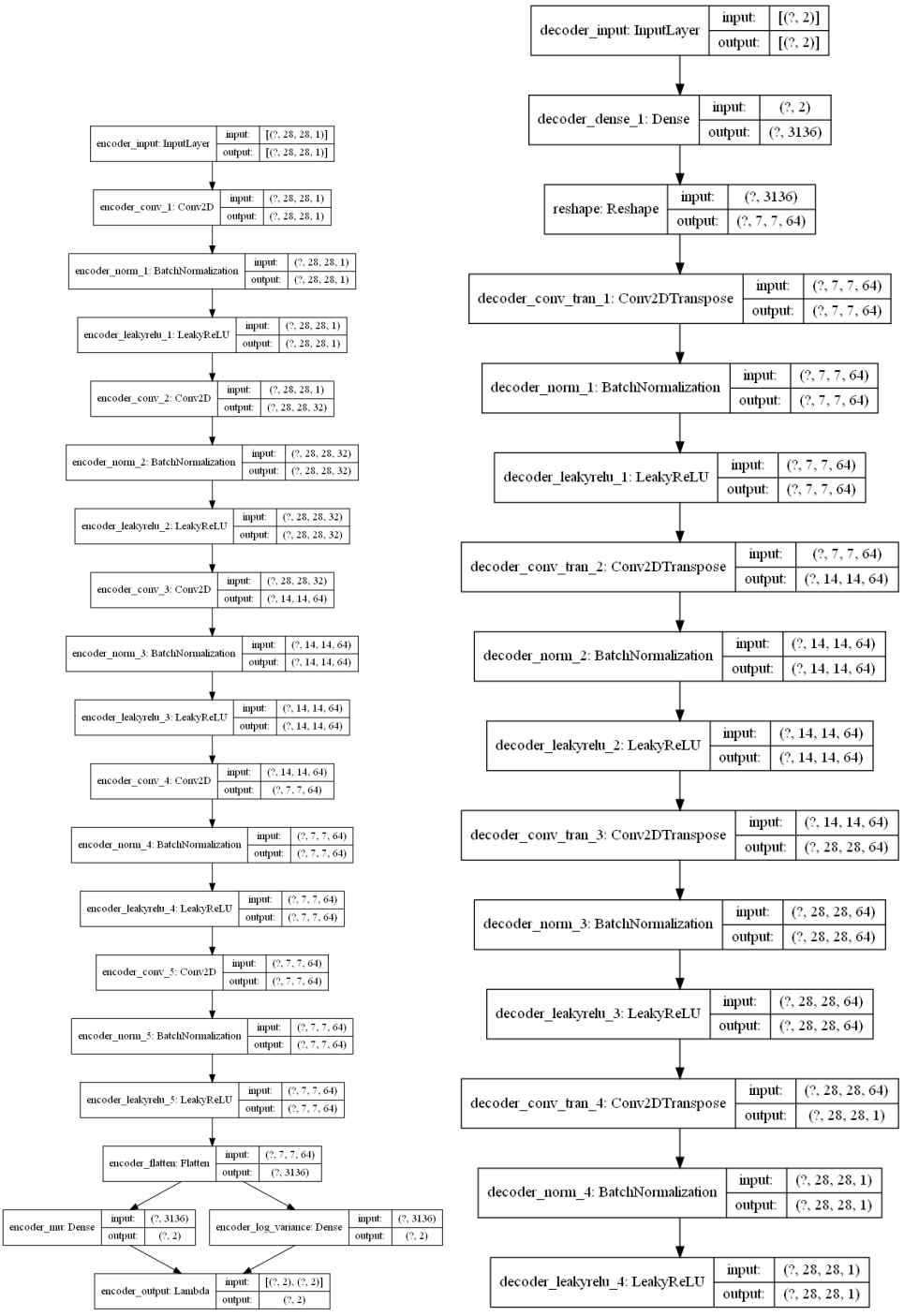
$$\min_G \max_D V(D,G) = E_{x \sim p_{data}(x)}[logD(x)] + E_{z \sim p_z(z)}[log(1 - D(G(z)))] \quad (1)$$

Where p_{data} and p_z represents the probability distribution of the real image and the generated image, respectively. During the training process of the discriminator, the objective function V is trained to minimize and maximize alternately, it is expected that the discriminant value of real data is as close to 1 as possible, and the discriminant value of generated data is as close to 0 as possible. In the process of training the generator, the gradient descent was used to minimize the value of the objective function. To illustrate, the structure of the DCGAN model implemented are shown as follow in Figure 1.



(a) DCGAN Generator (b) DCGAN Discriminator

Figure 1: The generator and discriminator of the DCGAN implemented



(a) VAE Encoder (b) VAE Decoder

Figure 2: The encoder and decoder of the VAE implemented

VAE is an important type of generative model, mainly used for image generation. It was proposed by Diederik P. Kingma and Max Welling in 2014. VAE improves the expression of the hidden layer on the basis of AE. In essence, the input image is efficiently encoded by the encoder E, and then the decoder D uses the encoding to restore the image[3]. Under ideal circumstances, the restored output image should be The original image is as close as possible.

In the VAE network. Encoder E receives the real image, performs feature encoding on it, and obtains the mean vector of feature encoding[3] $\mu = \{\mu^1, ...\mu^m\}$ and the variance vector $\sigma = \{\sigma^1, ...\sigma^m\}$. Then, combine with the Gaussian noise vector $\varepsilon = \{\varepsilon^1, ...\varepsilon^m\}$, we could have the implicit vector $Z = \mu + exp(\sigma) \cdot \varepsilon$. Decoding the implicit vector set, we could have $Y = f_\theta(Z) = \{y^1, ...y^m\}$, then we could minimize $J_{VAE} = E_{z-Q}[logP(x|z)] - KL[logQ(z|x)||P(z)]$, until we reach the maximum number of iterations.

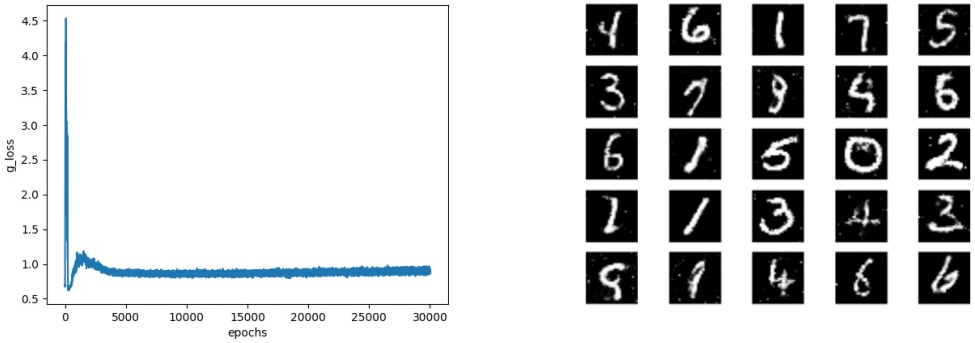
The structure of the VAE model implemented are shown as above in Figure 2.

3 Result and Comparison

This section would carry out the result of and a comparison between the GAN and VAE implemented. The data generated by the simple model at the beginning of the experiment is relatively fuzzy. After the optimization of the parameters and model structure, this problem has been improved.

3.1 GAN Result

The generated loss and generated images of the GAN implemented are shown below in Figure 3. 30000 epochs and 200 batch_size was used on 70000 images for training. Although the generated images seem quite decent, for further improvement, the node number of each layer and the layer numbers could be increased, and the learning rate to have a better result.

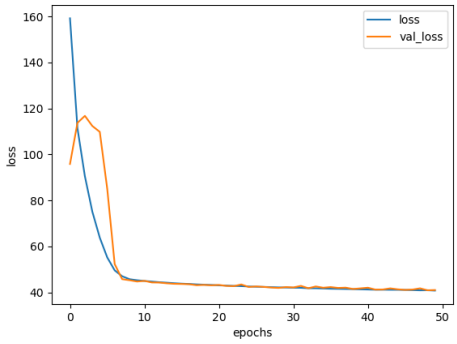


(a) GAN Generated Loss (b) GAN Generated Images

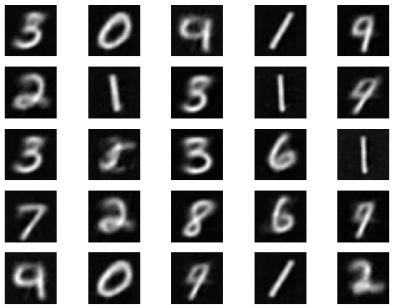
Figure 3: The generated loss and generated images of the GAN implemented

3.2 VAE Result

The generated loss and generated images of the VAE implemented are shown below in Figure 4. 200 epochs and 200 batch_size was used on 70000 images for training. The generated images are not as decent as the images that GAN generated, for further improvement, the node number of each layer and the layer numbers could be increased, and the learning rate to have a better result.



(a) VAE Training and Validating Loss



(b) VAE Generated Images

Figure 4: The training and validating loss and generated images of the VAE implemented

4 Conclusion

This paper implemented a DCGAN and a VAE model, carried out experiments on the MNIST handwritten digital data set, and compared the result generated. For future works, researchers could improve the model parameter and try implementing some more state-of-art model structures.

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. *In Advances in neural information processing systems* (pp. 2672-2680).

[2] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[3] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[4] Felix Mohr (2017). Implementing a Generative Adversarial Network (GAN/DCGAN) to Draw Human Faces. *https://towardsdatascience.com/implementing-a-generative-adversarial-network-gan-dcgan-to-draw-human-faces-8291616904a*

[5] fchollet (2020). Variational AutoEncoder. *https://keras.io/examples/generative/vae/*

[6] Joseph Rocca (2019). Understanding Variational Autoencoders (VAEs). *https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73*