Stabilized Training of Generative Adversarial Networks by a Genetic Algorithm

Hwi-Yeon Cho
Department of Computer Science, Kwangwoon Univ.
Seoul, Republic of Korea
hwyn.cho@gmail.com

Yong-Hyuk Kim
Department of Computer Science, Kwangwoon Univ.
Seoul, Republic of Korea
yhdfly@kw.ac.kr

ABSTRACT

Generative adversarial networks (GAN) facilitate the learning of probability distributions of complex data in the real world, and allow neural networks to generate the distribution. GANs (GAN and its variants) exhibit excellent performance in applications like image generation and video generation. However, GANs sometimes experience problems during training with regard to the distribution of real data. We applied a genetic algorithm to improve and optimize the GAN's training performance. As a result, the convergence speed and stability during the training process improved compared to the conventional GAN.

CCS CONCEPTS

• **Computing methodologies** → **Genetic algorithms**; *Neural networks*;

KEYWORDS

genetic algorithm, neural networks

ACM Reference Format:
Hwi-Yeon Cho and Yong-Hyuk Kim. 2019. Stabilized Training of Generative Adversarial Networks by a Genetic Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference 2019 (GECCO '19).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Generative adversarial networks (GAN) [1] constitute one of the major methods used for learning and generating complex data in the real world. To generate meaningful data that are similar to real data, GANs (GAN and its variants) adversarially train not only the generator but also the discriminator to classify real samples and generated fake samples. In recent years, such GANs have been successfully applied to applications like image generation and image editing.

However, when the distribution of the training dataset and the distribution of the generated samples are not similar, the GAN experiences issues with training. To solve this, deep convolutional GAN (DCGAN) [2] was developed to apply a convolutional structure to a conventional GAN that uses multilayer perceptron. Moreover, evolutionary GAN [3] was developed to apply evolutionary computation to the adversarial training. Specifically, generators generate offspring through mutation, and the population of generators evolves while preserving only the good generators.

We sought to improve and optimize the GAN's training performance by applying a GA. In particular, this was achieved by focusing on the oscillation phenomenon, in which the training is unstable and model does not converge even if trained for a significant amount of time. This study aims to improve the discrimination performance of the discriminator using the GA and to improve the generation performance of the generator. Experiments were conducted by applying the proposed method to the actual MNIST dataset. The results yielded performance improvements during the training process compared to the conventional GAN.

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 GENERATIVE ADVERSARIAL NETWORKS COMBINED WITH A GENETIC ALGORITHM

2.1 GAN

The GAN learns the minimax game between the generative network G and the discriminative network D. The G receives the latent vector $z \sim p(z)$ (sampled from a normal distribution) as input, and outputs new data $G(z) \sim p_g$ that approaches the data distribution p_{data}. On the other hand, the D discriminates the real data $x \sim p(x)$ and the data $G(z) \sim p_g(G(z))$ generated by G. Such a training process for the GAN can be expressed by the following equation.

$$\min_G \max_D \mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log (1 - D(G(z)))] \quad (1)$$

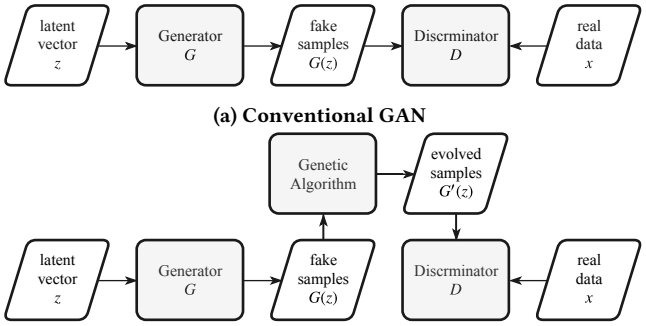
If the training is continued in this manner, then, at the end, $p_{data} = p_g$, which represents the state in which the D cannot discriminate between the real and the fake, i.e., $D(x) = D(G(z)) = \frac{1}{2}$.

2.2 Parameters of GA

A population of the same size as the GAN's training batch size was chosen, and each chromosome was a fake sample (i.e., $G(z)$) generated by the G. The fitness of the chromosome was the D's discrimination value (i.e., $D(G(z))$). Roulette wheel selection was used for the selection type, and arithmetic recombination crossover for the crossover type. The mutation was carried out by adding an arbitrary number between -0.05 and $+0.05$ to each gene with a probability of 10%. When the fitness of the offspring was higher than that of the individual with the lowest fitness of the conventional population, the worst individual was replaced by the offspring.

2.3 GAN combined with GA

A conventional GAN trains the D and G alternately, as shown in Figure 1(a). Because of this, at a certain step, a problem may occur



(a) Conventional GAN

(b) GAN combined with GA

Figure 1: Conventional GAN and the proposed GAN

whereby the training of the D and the training of the G cancel each other out. By applying the GA to each training step, as shown in Figure 1(b), we attempted to improve the discrimination ability of the D and, accordingly, improve the performance of the G.
First, the initial population was populated with the fake samples (i.e., chromosome) generated by the G, and the fitness of each fake sample was calculated. The samples of high fitness (samples that were discriminated as real by the D) were selected, and the population was evolved by using the parameters introduced in Section 2.2. By supplying it with fake samples from the evolved population (i.e., $G'(z)$), the D was trained. Through this process, the D should be perform better, in turn causing the G to improve even more.

3 RESULTS AND ANALYSIS

In this section, the experiments and analysis are discussed for tasks of several generations to evaluate the proposed GAN combined with GA. The data used in the experiments were from the MNIST dataset (a dataset that consists of handwritten digits between 0 and 9), which are widely used for training and validation in machine learning. We trained the GAN and performed the data generation experiment by separating the data by each digit. The performance was measured using the loss (binary cross-entropy) of the discriminator and the generator before and after the application of the GA to the conventional GAN. The binary cross-entropy is calculated from the equation below. When the GAN is trained and $D(x) = D(G(z)) = \frac{1}{2}$ is obtained, the loss value becomes 0.69, at which point model has converged.

$$H[X] = -[P(X=0)\log P(X=0) + P(X=1)\log P(X=1)] \quad (2)$$

Figure 2 compares the loss of the discriminator and the generator before and after application of the GA to the GAN. Although these did not converge before or after application of the GA to the GAN, the training proceeded more stably and more convergently when the GA was applied. Figure 3 shows the kernel density estimate plot for the distribution of digit 0 and the distribution of samples generated during the training of the GAN. For each step, the data generated by the GAN combined with GA closely matched the distribution of the actual data, and the training and generation performance improved slightly.

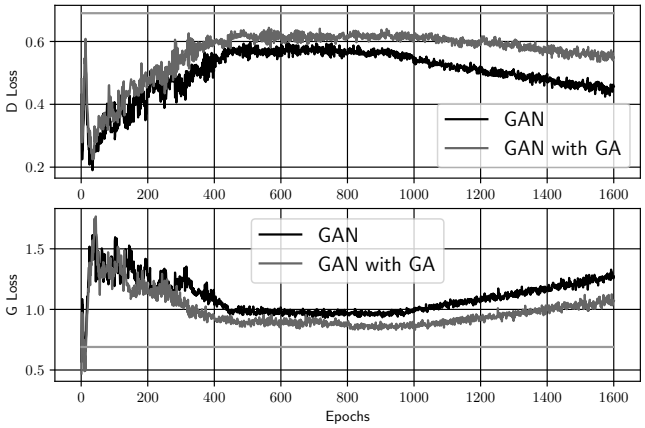
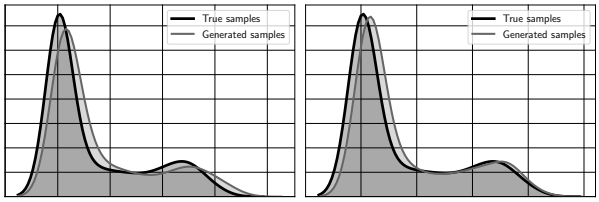


Figure 2: Comparison of loss for each step of GAN during training for the digit 0



(a) Conventional GAN **(b) GAN combined with GA**

Figure 3: Comparison of generation performance with GA at step 420

4 CONCLUSION

GANs constitute one of the major methods used in data generation and editing. Additionally, the GA approach is widely used in various optimization problems. This study proposed an approach in which a GA was utilized to address the GAN's issues and improve performance. When the GA was used, the generated samples evolved in a diverse manner and the discriminator was able to discriminate more stably. Although there was no noticeably large improvement, it was confirmed that it is possible to solve some of the issues present in GANs and improve performance by applying a GA. In the future, we plan to conduct a more in-depth study on methods for improving GANs by applying a GA and improving the generation performance.

REFERENCES

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*. 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets

[2] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR* abs/1511.06434 (2015). arXiv:1511.06434 http://arxiv.org/abs/1511.06434

[3] Chaoyue Wang, Chang Xu, Xin Yao, and Dacheng Tao. 2018. Evolutionary Generative Adversarial Networks. *CoRR* abs/1803.00657 (2018). arXiv:1803.00657 http://arxiv.org/abs/1803.00657