Applying Evolutionary to Generative Adversarial Networks (May 2020)

M. C. Nguyen, H. D. Nguyen , *Sophomore*

†*Neural Network and Genetic Algorithm, CS410.K21.KHCL*

†*University of Information Technology, Department of Computer Science*

18520519@gm.uit.edu.vn, 18520606@gm.uit.edu.vn

***Abstract*——— Generative Adversarial Networks (GAN) are one of the most interesting and popular applications of Deep Learning. Nevertheless, it has been faced some training problems such as mode collapse, non-convergence, diminished gradient, even its variants cannot completely overcome these problems. It matters a great deal when we stop training the networks. These problems mainly arise from a lack of diversity in their adversarial interactions. Therefore, we introduce a technique that mimics the steps of evolution in nature called Evolutionary Generative Adversarial Networks (E-GAN) applying the principles of evolutionary computation to mitigate these problems to improve the limitations of the existing GAN, its stability and also generative performance. The population of generators evolves in a dynamic environment, in this case, the discriminator. Each evolutionary step consists of three sub-stages: variation, evaluation, and selection. All of the best offspring will be kept and develop the population to adapt to the environment, we also combine different adversarial training objectives as mutation to diversify the generative samples. If this stimulation plays over a long time, the resulting algorithm can be trained more stably. Finally, we will represent the difference between E-GAN and basic GAN by training on some datasets to demonstrate the achievement of E-GAN on enhanced performance and reducing the training problems.**

***Index Terms* — Generative adversarial networks, Evolutionary GAN.**

I. Introduction

GAN have many successes in generating realistic, complex distributions and can be used in many applications such as image generation, video prediction, image in-painting, and text to image synthesis.

However, GAN have a lot of problems such as mode collapse, vanishing gradients, or non-convergence. Therefore, it is hard to train a model by original GAN. There are many variants of GAN created to overcome GAN issues. The common way to deal with these problems is by changing the objective function is the one used to measure the difference between real data distribution and generated data distribution. The original GAN uses Jensen-Shannon divergence as the metric. Other measures that can be used include Kullback-Leibler divergence, Wasserstein distance used in TF-GAN, the least-squares, the absolute deviation. Although there are many variants of GAN that fix the issues they still have some weaknesses.

To take all the advantages of different metrics and also suppress the weaknesses for each of them, Evolutionary generative adversarial network (E-GAN), which is use the evolution technique in GAN. Specifically, a discriminator acts as the environment because it provides loss functions and they can provide the fitness for individuals in population of generators so they can gradually evolves. During each iteration, the discriminator is still trained to recognize real and fake samples. However, generators use different mutations to produce offspring. Different objective functions aim to minimize different distances between the generated distribution and the data distribution, leading to different mutations. Meanwhile, given the current optimal discriminator, we measure the quality and diversity of samples generated by the updated offspring. Finally, according to the principle of "survival of the fittest", poorly-performing offspring are removed and the remaining well-performing offspring are preserved and used for further training.

II. Related Works

In this section, we first analyze some previous GAN devoted to reducing training instability and improving generative performance. We then shortly summarize some evolutionary algorithms on deep neural networks.

2.1 Generative Adversarial Networks

Generative adversarial networks (GAN) is a topic of enormous recent interest, providing a powerful class of tools for the unsupervised learning of probability distributions over difficult manifolds such as natural images for learning deep generative models. Compared to other generative models, GAN is easily trained by alternately updating a generator and a discriminator using the back-propagation algorithm. In many generative tasks, GAN (GAN and its variants) produce better samples than other generative models.

However, the original GAN model has many problems. In the original GAN, if the discriminator is too good, then generator training can fail due to vanishing gradients. In effect, an optimal discriminator doesn't provide enough information for the generator to make progress. To solve this issue, the original GAN paper proposed a modification to minimax loss to deal with vanishing gradients. Second, each iteration of generator over-optimizes for a particular discriminator and the discriminator never manages to learn its way out of the trap. As a result, the generators create a small set of output types. This form of GAN's failure is called **mode collapse**. To solve this issue, try to force the generator to broaden its scope. The Wasserstein loss alleviates mode collapse by letting you train the discriminator to optimality without worrying about vanishing gradients. If the discriminator doesn't get stuck in local minima, it learns to reject the outputs that the generator stabilizes on. So, the generator has to try something new.

2.2 Evolutionary Algorithms

Among the set of search and optimization techniques, the development of Evolutionary Algorithms (EA) has been very important in the last decade. Inspired by natural evolution, the essence of an evolutionary approach to solve a problem is to equate possible solutions for individuals in a population, and to introduce a notion of fitness based on solution quality. To obtain a working evolutionary algorithm one has to go through a number of design steps. The first step is to identify a representation: that is a suitable data structure that can represent possible solutions to the problem. The next step is to define a way of measuring the quality of an individual based on problem specific requirements. The final step is to specify suitable selection and variation operators.

Recently, evolutionary algorithms have been introduced to solve deep learning problems. To minimize human participation in designing deep algorithms and automatically discover such configurations, there have been many attempts to optimize deep learning hyper-parameters and design deep network architectures through an evolutionary search Evolutionary algorithms have also demonstrated their capacity to optimize deep neural networks .Last but not least, an evolutionary algorithm was proposed to compress deep learning models by automatically eliminating redundant convolution filters.

III. Method

3.1 Generative Adversarial Networks

Generative Adversarial Networks (GAN) is the main method for generating models from complicated data. GAN consist of two neural networks: Discriminator and Generator. The input of the generator is the random noise and the output of the generator is the image corresponding to the random input. Then the output of the generator will be used as the input of the discriminator in order to be considered if it is real or fake by the discriminator. The discriminator will give the prediction and it will be taken to two loss functions to update discriminator and generator.

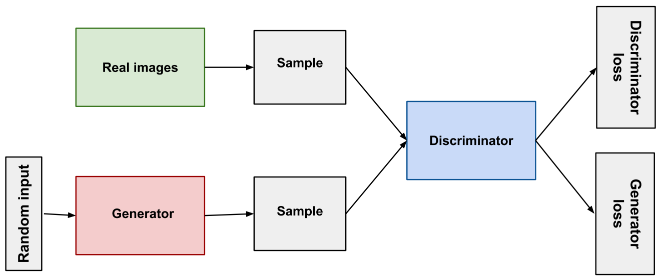


Figure 1: Model of Generative Adversarial Networks.

Basically, the discriminator is trying to distinguish the real data and the fake one created by the generator. On the other hand, as a confronter, the generator aims to fool the discriminator by creating more realistic samples. The training continues until the discriminator cannot recognize the fake data anymore. That means the generator wins the adversarial game.

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

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

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.

3.2 Evolutionary Algorithm

A conventional GAN trains the D and G alternately, as shown in Figure 2 (a) by applying the evolutionary algorithm that evolves a population of generator(s) {G} in a given environment (the discriminator D). In this population, each individual represents a possible solution in the parameter space of the generative network G. During the evolutionary process, we expect that the population gradually adapts to its environment, which means that evolved generator(s) can generate ever more realistic samples and finally learn the real-world data distribution.

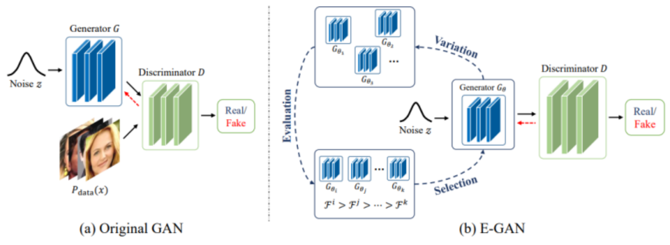


Figure 2: Conventional GAN and the proposed GAN.

As shown in Figure 2 (b), during evolution, each step consists of three sub-stages:

- **Variation:** Given an individual G_θ in the population, we use the variation operators to produce its offspring. Specifically, several copies of each individual or parent are created, each of which is modified by different mutations. Then, each modified copy is deemed as one child.
- **Evaluation:** For each child, its performance or individual's quality is evaluated by a fitness function F(.) that depends on the current environment (discriminator D).
- **Selection:** To fitness value, all individual children will be selected and the worst element is removed. The rest remain alive and evolve to the next iteration.

Alter each evolutionary step, the discriminator is updated to further distinguish real samples x and fake samples y generated by generator.

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\mathrm{data}}}[\log D(x)] - \mathbb{E}_{y\sim p_g}[\log(1 - D(y))]. \quad (2)$$

Therefore, the discriminative network D (i.e., the environment) can continuously provide the adaptive losses to push the population of generator(s) evolving to produce better solutions. Next, we illustrate and discuss the proposed variation (or mutation) and evaluation operators in detail.

3.3 Mutation

To produce individuals for the next generation, EGAN uses three different mutations, each mutation has distinct training objectives, and it selects the individual that provides the best fitness in order to against the discriminator.

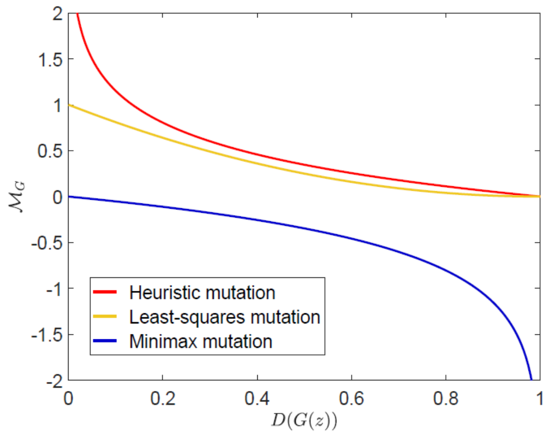


Figure 3: The mutation functions that generator receives by discriminator.

3.3.1 Minimax mutation:

The minimax mutation corresponds to the minimax objective function in the original GAN:

$$\mathcal{M}_G^{\mathrm{minimax}} = \frac{1}{2}\mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]. \quad (3)$$

According to the original paper, to calculate the distance between the data distribution and the generated distribution, they use the Jensen-Shannon divergence (JSD). It calculates how different two or more distributions are from each other. However, there is a problem for this is that when the discriminator rejects generated samples with high confidence (D(G(z)) → 0 then the log function will become 0), the gradient tends to vanish. leading to the vanishing gradient. This problem is visually illustrated in Figure 3. However, if the generated distribution nearly overlaps with the data distribution, meaning that the discriminator cannot completely distinguish real from fake samples, the minimax mutation provides effective gradients and continually narrows the gap between the data distribution and the generated distribution.

3.3.2 Heuristic mutation:

To avoid the problem of the minimax mutation, which is trying to minimize the log probability, the heuristic mutation tends to maximize the log probability:

$$\mathcal{M}_G^{\text{heuristic}} = -\frac{1}{2}\mathbb{E}_{z\sim p_z}[\log(D(G(z))]. \quad (4)$$

Compared to the minimax mutation, the heuristic mutation will not saturate. When the discriminator rejects the generated samples then the objective function will not become 0. Thus, the heuristic mutation avoids vanishing gradient and provides useful generator updates in Figure 3. However, in practice, this may lead to training instability and generative quality fluctuations.

3.3.3 Least-squares mutation:

The least-squares mutation is inspired by LSGAN, where the least-squares objectives are utilized to penalize its generator to deceive the discriminator. In this work, we formulate the least-squares mutation as:

$$\mathcal{M}_G^{\text{least-square}} = \mathbb{E}_{z\sim p_z}[(D(G(z)) - 1)^2]. \quad (5)$$

The least-squares mutation has some similarities to the heuristic mutation which is the non-saturating when the discriminator can recognize the generated sample. When the discriminator output rises, the least-squares mutation saturates, gradually approaching zero. Therefore, the least-squares mutation can avoid vanishing gradient when the discriminator has a significant advantage over the generator. Meanwhile, compared to the heuristic mutation, although the least-squares mutation will not assign an extremely high cost to generate fake samples, it will also avoid mode collapse.

3.4 Evaluation

In an evolutionary algorithm, evaluation is used to measure the quality of individuals. To decide on which individuals' selection, we devise an evaluation function to measure the performance of evolved individuals (children). Specifically, we concentrate on two generator properties: 1) the quality and 2) the variety of generated samples. First, the generator feeds into the discriminator net and observe the average value of the output, which we name the *quality fitness score*:

$$\mathcal{F}_{\text{q}} = \mathbb{E}_z[D(G(z))]. \quad (6)$$

In the training process, the discriminator D is continuously updated to be optimal, reflecting the quality of generators at each evolutionary (or adversarial) step. By this way, the generator improves with training (high-quality score), the discriminator performance gets worse because the discriminator can't easily tell the difference between real and fake.

Not only interested in generative quality but we also pay attention to the diversity of generated samples. There are many suggestions about using the gradient to improve the stability and optimize GAN and reduce the probability of mode collapse. By using this method, when the generator creates images that are nearly the same as the previous ones which have a high prediction of the real image from the discriminator, then they will seem as fake images from the discriminator. That means if the discriminator is still making a wrong prediction that leads to the mode collapse will be punished by using the formula:

$$\mathcal{F}_{\text{d}} = -\log\left|\left|\nabla_D - \mathbb{E}_x[\log D(x)] - \mathbb{E}_z[\log(1 - D(G(z)))]\right|\right|. \quad (7)$$

The log gradient value of the discriminator will be used for the diversity of the generated images. If the generator creates images with high diversity then discriminator gradients will be small. Thus, the mode collapse problem will be solved and the discriminator becomes more stable during the training.

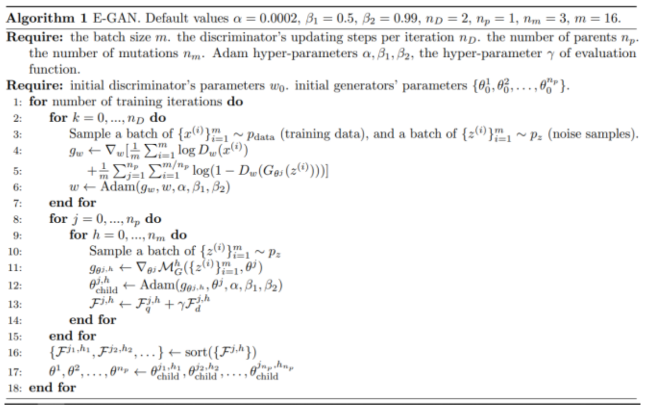
From two formula for calculating the quality and diversity fitness, we have the final formula for evaluating the individual's fitness:

$$\mathcal{F} = \mathcal{F}_{\text{q}} + \gamma\mathcal{F}_{\text{d}} \quad (8)$$

Lambda >= 0 to balance two metric: quality and diversity.

3.5 E-GAN

The complete E-GAN training process is summarized in Algorithm 1. In E-GAN, discriminator acts as the environment and a population of generators evolves in response to the environment. For each evolutionary step, generators undergo different mutations to produce offspring to adapt to the environment. According to the principle of "survival of the fittest", the remaining well-performing offspring are preserved and used for further training. E-GAN algorithm to combine the advantages of different adversarial objectives and generate the most competitive solution. Consequently, during training, the evolutionary algorithm not only mostly overcomes the restrictions (vanishing gradient, mode collapse, etc.) of individual adversarial objectives, but it also provides their benefits to search for a better solution.



Algorithm 1 E-GAN. Default values $\alpha = 0.0002$, $\beta_1 = 0.5$, $\beta_2 = 0.99$, $n_D = 2$, $n_p = 1$, $n_m = 3$, $m = 16$.

Require: the batch size m. the discriminator's updating steps per iteration n_D. the number of parents n_p. the number of mutations n_m. Adam hyper-parameters α, β_1, β_2, the hyper-parameter γ of evaluation function.

Require: initial discriminator's parameters w_0. initial generators' parameters $\{\theta_0^1, \theta_0^2, \ldots, \theta_0^{n_p}\}$.

```
1:  for number of training iterations do
2:      for $k = 0, ..., n_D$ do
3:          Sample a batch of $\{x^{(i)}\}_{i=1}^m \sim p_{\text{data}}$ (training data), and a batch of $\{z^{(i)}\}_{i=1}^m \sim p_z$ (noise samples).
4:          $g_w \leftarrow \nabla_w[\frac{1}{m}\sum_{i=1}^m \log D_w(x^{(i)})$
5:              $+\frac{1}{m}\sum_{j=1}^{n_p}\sum_{i=1}^{m/n_p} \log(1 - D_w(G_{\theta^j}(z^{(i)})))]$
6:          $w \leftarrow \text{Adam}(g_w, w, \alpha, \beta_1, \beta_2)$
7:      end for
8:      for $j = 0, ..., n_p$ do
9:          for $h = 0, ..., n_m$ do
10:             Sample a batch of $\{z^{(i)}\}_{i=1}^m \sim p_z$
11:             $g_{\theta^{j,h}} \leftarrow \nabla_{\theta^j}\mathcal{M}_G^h(\{z^{(i)}\}_{i=1}^m, \theta^j)$
12:             $\theta_{\text{child}}^{j,h} \leftarrow \text{Adam}(g_{\theta^{j,h}}, \theta^j, \alpha, \beta_1, \beta_2)$
13:             $\mathcal{F}^{j,h} \leftarrow \mathcal{F}_q^{j,h} + \gamma\mathcal{F}_d^{j,h}$
14:         end for
15:     end for
16:     $\{\mathcal{F}^{j_1,h_1}, \mathcal{F}^{j_2,h_2}, \ldots\} \leftarrow \text{sort}(\{\mathcal{F}^{j,h}\})$
17:     $\theta^1, \theta^2, \ldots, \theta^{n_p} \leftarrow \theta_{\text{child}}^{j_1,h_1}, \theta_{\text{child}}^{j_2,h_2}, \ldots, \theta_{\text{child}}^{j_{n_p},h_{n_p}}$
18: end for
```

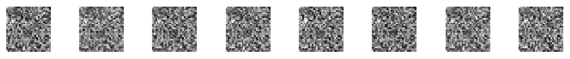
IV. Experiment

We implement EGAN on dataset MNIST to see the performance of EGAN with table of network topology for dataset MNIST:

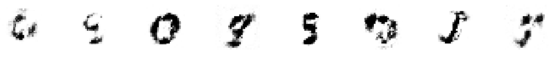
Parameter	Value
Input	100
Number of hidden layers	3
Output	784
Activation function	Sigmoid
Batch size	32
Stop condition	2 hours
Survived children	1
Optimizer	Adam
Learning rate for Adam G	0.0002
Learning rate for Adam D	0.0002
Hyperparameter for fitness function	0.002

After 2 hours, we get the result as the image below for each 5 epochs:

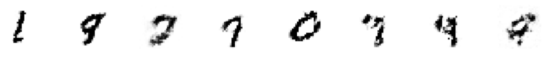
Epoch 0:



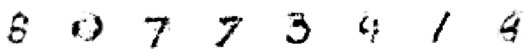
Epoch 5:



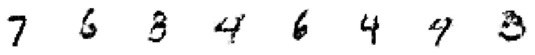
Epoch 10:



Epoch 15:



Epoch 20:



V. Conclusion

We propose an evolutionary GAN framework (E-GAN) for training deep generative models. When the evolutionary algorithm was used to evolve a population of generators to adapt to the dynamic environment (the discriminator D), which will prevent the original GAN's problem and improve generator performance.

VI. Future Works

In the future, we will improve EGAN implementation on MNIST dataset to give more stable performance. We also try to implement EGAN on other datasets like Cifar10, bedroom, e.t.c.

VII. References

[1] Chaoyue Wang, Chang Xu, Xin Yao, Dacheng Tao, Evolutionary Generative Adversarial Networks, arXiv:1803.00657 (2018).

[2] Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, arXiv preprint arXiv:1701.07875 (2017).

[3] Ian J. Goodfellow, Generative Adversarial Nets, NIPS (2014).

[4] Hwi-Yeon Cho, Yong-Hyuk Kim ,Stabilized Training of Generative Adversarial Networks by a Genetic Algorithm, doi.org/10.1145/3319619.3326774.

[5] Jamal Toutouh , Erik Hemberg, Una-May O'Reilly, Spatial Evolutionary Generative Adversarial Networks, arXiv:1905.12702v1

[6] K. A. De Jong. Evolutionary computation: a unified approach. MIT press, (2006).

[7] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. In Proceedings of the International Conference on Learning Representations (ICLR), (2017).