Project Report

GA-GAN: a genetic algorithm based generative adversarial network

Yiqiang Zhou 101177158

Abstract

A new generative adversarial network is developed to generate new doodle images from existing datasets. Unlike the traditional generative adversarial networks, in which both generator and discriminator are artificial neural networks[1], this project uses genetic algorithm as generator. In genetic algorithm, a population of candidate solutions and a fitness evaluation are needed to evolve towards better solutions, thus the discriminative network is used to evaluate the fitness of each individuals in the population. A CNN based GAN is also developed as a comparison with the GA-GAN. Experiments demonstrate that this type of GAN has the basic ability to produce doodle images, but has difficulty in producing more complicated images.

1. Introduction

All kinds of generative adversarial networks have been developed to solve various problems[2]. GANs have been used for semi-supervised learning[2], fully supervised learning[4], and reinforcement learning[5]. This type of machine learning systems is capable of learning the patterns from input data and generate new data that is alike the original dataset. It's still a relatively generative model, so this project tries to propose a different kind of generator to see if the new adversarial model can perform better or if worse, why.

The dataset this project choose to learn from is the "world's largest doodling data set[6]" assembled and preprocessed by Google. It's a dataset of doodles drawn by people from all around the world with categories attached. This project chooses three categories, camel, star, and circle.

In the following sections, the report will firstly introduces the essential background of this project, including convolutional neural network, deconvolutional neural network generative adversarial network and genetic algorithm. Then it will briefly describes the

related work in the area. In the methodology section, the algorithms and models used by this project will be described as well as the dataset it uses. In the discussion section, the pros and cons of this model will be discussed, and some methods adopted to improve the result will be inspected and discussed too. Finally, in the conclusions and future work section, a short conclusion will be drawn and possible ways to improve the result in the future are proposed.

2. Background

Convolutional Neural Network

A convolutional neural network is a class of deep neural networks that use convolutional layers. Those layers share weights thus reduce parameter number. It has space invariance, and is well suited for image analysis. This project uses convolutional neural network as discriminator.

Deconvolutional Neural Network

A deconvolutional neural network is a class of deep neural networks that use transposed convolutional layers to upsample the input to a larger size. By using different padding and striding methods, a deconvolutional layer (or a transposed convolutional layer) increases the size of the input. This project uses a DNN as a generator to confirm the discriminator can successfully distinguish fake data from the real data and to make a comparison of DNN as generator to the GA generator.

Generative Adversarial Network

A generative adversarial network (GAN) is a class of machine learning system that have two adversarial parts: generative model and discriminative model. These two parts contest with each other in the process of training. The generator tries to produce data that increases the loss or decrease the accuracy of the discriminative model while the discriminator tries to correctly classify the data produced by the generator and the real data. Usually these two models are all artificial neural networks, and in a popular GAN model, the generator is DNN and the discriminator is CNN. It's called "DCGAN"[7] (Deep convolutional generative adversarial networks).

Genetic Algorithm

A genetic algorithm (GA) is a metaheuristic that inspired by some ideas from natural selection to solve optimization problems. It usually requires a genetic representation of the solution and a fitness function to evaluate the solution. In addition to that, it requires a method to select individuals from the population and ways to perform genetic operators.

3. Related Work

The original paper about GAN is published in 2014[8]. It is in this paper that the idea "adversarial" is introduced in a generative model.

Some traditional GANs based on the google quickdraw dataset have been developed[9] [10]. And some researches have been done trying to combine GA and GAN, but most of them are using GA to tune the parameter of the generative network[11], not using genetic algorithm itself as the generator.

And some articles described the limitations and possible reason why a GAN may not work and the ways to resolve them[12][13].

4. Methodology

Dataset

This project uses *Quick, Draw! Dataset*[14], "a collection of 50 million drawings across 345 categories, contributed by players of the game Quick, Draw!." (See Figure 1) The doodles has been preprocessed by google into different format. This project uses the numpy bitmap files, which are 28x28x1 numpy arrays represent different doodles in pixels.

Figure 1. Quick Draw! Dataset

For this project, mainly three categories are used: camel, circle and star.

DCGAN

In order to make sure the discriminator used in the project has the potential to distinguish fake data from the real ones, this project firstly build a more studied GAN, using deconvolutional NN as generator and discriminator.

The architecture of the generative model and discriminative model are shown as below:

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 12544)	1254400
batch_normalization (BatchNo	(None, 12544)	50176
leaky_re_lu (LeakyReLU)	(None, 12544)	0
reshape (Reshape)	(None, 7, 7, 256)	0
conv2d_transpose (Conv2DTran	(None, 7, 7, 128)	819200
batch_normalization_1 (Batch	(None, 7, 7, 128)	512
leaky_re_lu_1 (LeakyReLU)	(None, 7, 7, 128)	0
conv2d_transpose_1 (Conv2DTr	(None, 14, 14, 64)	204800
batch_normalization_2 (Batch	(None, 14, 14, 64)	256
leaky_re_lu_2 (LeakyReLU)	(None, 14, 14, 64)	0
conv2d_transpose_2 (Conv2DTr	(None, 28, 28, 1)	1600

Total params: 2,330,944
Trainable params: 2,305,472
Non-trainable params: 25,472

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 14, 14, 64)	1664
dropout (Dropout)	(None, 14, 14, 64)	0
conv2d_1 (Conv2D)	(None, 7, 7, 128)	204928
dropout_1 (Dropout)	(None, 7, 7, 128)	0
conv2d_2 (Conv2D)	(None, 4, 4, 256)	819456
dropout_2 (Dropout)	(None, 4, 4, 256)	0
conv2d_3 (Conv2D)	(None, 4, 4, 512)	3277312
dropout_3 (Dropout)	(None, 4, 4, 512)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 1)	8193

Total params: 4,311,553
Trainable params: 4,311,553
Non-trainable params: 0

The result shows this GAN model is able to produce plausible doodles after about 5000 epochs.

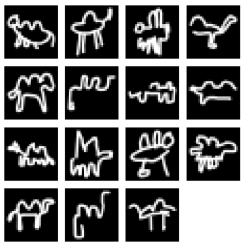

Figure 2. Preview of the original dataset

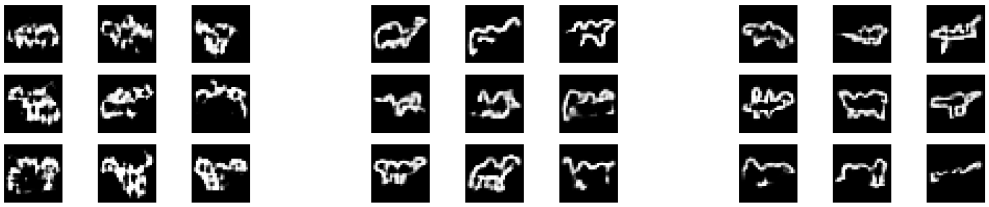

Figure 3. Result produced by the DCGAN after 1000, 3000 and 5000 epochs

The same discriminator is used in the following model.

GA-GAN

Overall algorithm

In this GAN model, a genetic algorithm model is used as generator. A simplified version of algorithm can be described as the following:

```
Initialize the population (a random set of images fo size N)
Repeat N times
      Use the CNN discriminator to evaluate each solution

      randomly select a batch B1 from population
      generate a random image batch B2
      randomly select a batch B3 from real data
      Train the CNN model on random selected batches [B1, B2, B3]

      Tournament selection and elite selection
      crossover and mutation
End repeat
```

Algorithm 1. Simplified Version

There are few things worth mentioning in the algorithm. First, a tournament selection (Figure 4) strategy is used. It's because if selecting best individuals as parent every generation, the model loses its diversity quickly thus converge too early. This is also known as mode collapse problem in GAN[15].

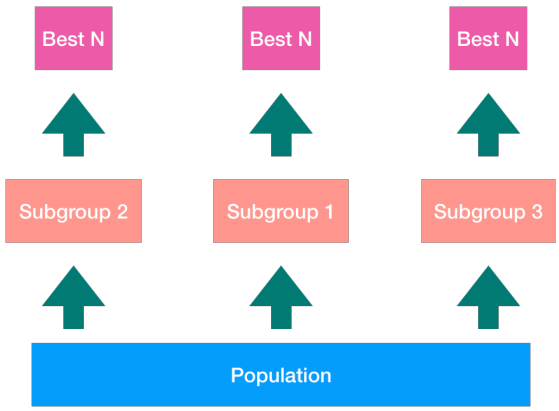

Figure 4. Tournament selection

The best one individual is also chosen in every generation to ensure the one with best fitness evaluation will be preserved. Some randomly generated data is used to be fetched to the discriminator, in case it learns from some specific feature from the population too effectively.

But it's inefficient to successfully train the model using this algorithm, because of another major problem when training GAN: diminished gradient. The discriminator is too capable of classifying the data, so after the first few epochs, the discriminator becomes too powerful. Almost every individual is rated a very low fitness (close to 0). The genetic algorithm thus is not able to select the better individuals effectively since they are all very alike. As the process goes on, the generator and discriminator become more imbalanced. To resolve this problem, every time before training the CNN, it will evaluate the accuracy of that discriminative model, if it is higher than a certain value, the training process in this epoch is skipped.

Representation

The representation of the image in the genetic algorithm is very important. At first, a 28x28 bit numpy array is used. Though it's an intuitive representation, it failed to successfully produce the least acceptable images even after 500 epochs possibly because of the overlarge search space (2^{28x28}).

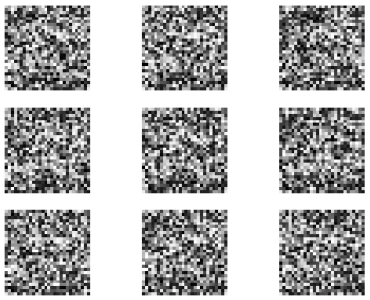

Figure 5. result of the 28x28 bit array representation

To resolve this problem, a line-based representation is used. It's a 28x28x28x28 bit array. The four dimensions represent the two 2D-points that a line passes through. Although the search space is seemingly lager ($2^{28x28x28x28}$), the meaningful subset ratio is increased because a human-drawn doodle is naturally composed of lines.

Genetic operations

Using this representation, the crossover and mutation operations are defined as below.

```
Select a random number x between 1 and 28 (image height)
Exchange all lines whose start points are above height x of image A
with all lines whose start points are above height x of image B
```

Algorithm 2. Crossover Operation

The reason crossover operation is implemented in this way is mainly because of the simplicity of the algorithm.

```
If mutate
      For every existing line
            If random number < p1
                  remove that line
            End if

            If random number < p2
                  change the ends of this line slightly
            End if
      End for

      If random number < p3
            Invert the connection between two random points
      End if
End if
```

Algorithm 3. Mutate operation

Result

Using population size of 256, tournament size of 10, crossover rate of 0.8 and mutation rate of 0.1, trained for 2000 epochs, the result is:

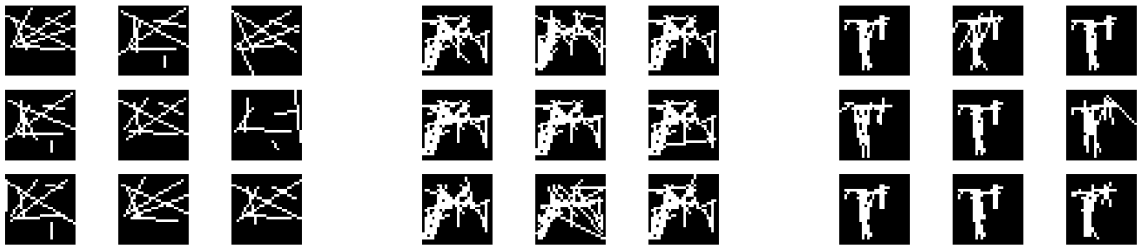

Figure 6. Train result (camel)

It can be seen that the genetic algorithm is unable to produce recognizable camels.

If uses star and circle as dataset, more meaningful result can be produced:

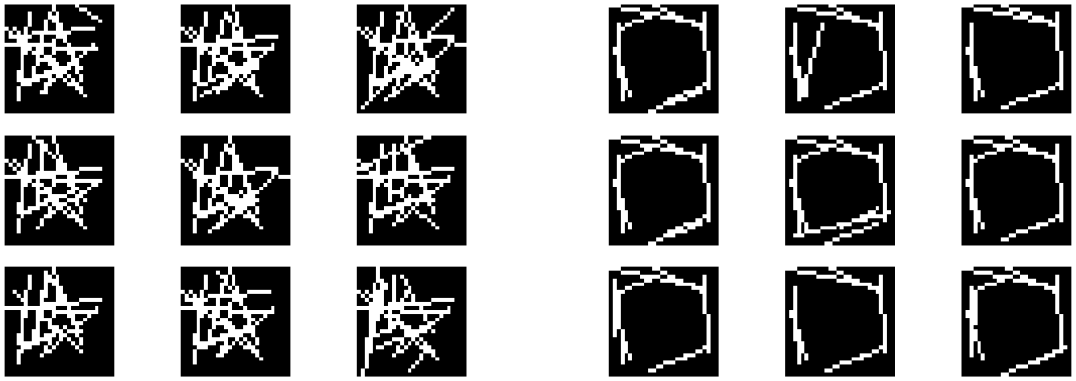

Figure 7. Train result (star and circle)

The possible reasons are discussed in the next section.

5. Discussion

In this project, a genetic algorithm based generative adversarial network is proposed and can successfully produce simple doodles like star and circle. Some techniques have been adopted to address the problems occurred during the training. The tournament selection is used to remedy early coverage (in genetic algorithm) or mode collapse (in genetic adversarial network). Some random noises are added as part of the training batch to reduce the possibility that discriminator learns from some specific features owned by the generator. And a “train only if inaccurate” method is used to address the problem where discriminative model learns too fast. Thus the model can

successfully produce simple doodles. However, it still failed to produce more complicated image like camel.

The possible reason of this result may be that the search space is still too large, therefore the probability that a genetic algorithm finds the “right” lines is too low. It may also because the genetic operations are not so well-suited for this problem.

Another possible reason in a deeper level is that, while the GAN requires the generator and discriminator work in balance, the genetic algorithm is too slow when finding the optimal value compared with the CNN, even if a method has been adopted to remedy this problem. And also, in the case of DCGAN, where images are successfully produced, the generator and discriminator are trained equally continuously. That is to say, on each batch, gradient descents are performed both on generator and discriminator, so the changes of their weights are continuous, and more importantly, it to a degree guarantees to improve the result, and to coverage in the end. But in genetic algorithm, when tournament selection is not used, it converge too early, but when it is used, the GA failed to retain the the most promising results as the discriminator continues to improve. It can be proven by the fact that after a circle like shape has already been produced by GA, it may still produce worse result in the following epochs.

Another observation is that the representation is so important that before changing it, it’s impossible to produce any image, but after changing it, it immediately becomes possible. Also many researches have been done using genetic algorithm to find the optimal weights of a neural network. So maybe GA is not so competent when dealing with low-level representation but more useful when a higher-level representation is used or when tuning the parameters of some other model.

6. Conclusions and Future Work

This project introduces a genetic algorithm based generative adversarial network, unlike normal GANs, it uses genetic algorithm as generator. It’s able to generate new doodles of simple shape from existing datasets, but still has difficulty in generating more complex object. It can also be observed that this GA-GAN has problem to converge at the end. Because even if it has already produce fairly good result in the case of circle, it may still produce worse result in the following epochs. So the future work might be find a way to let the model converge more stably. Or try use a more advanced representation of the data and let the genetic algorithm find the optimal parameters of that representation instead of the raw data.

References

[1] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). *Generative Adversarial Networks* (PDF). Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

[2] Caesar, Holger (2019-03-01), A list of papers on Generative Adversarial (Neural) Networks: nightrome/really-awesome-gan

[3] Salimans, Tim; Goodfellow, Ian; Zaremba, Wojciech; Cheung, Vicki; Radford, Alec; Chen, Xi (2016). "Improved Techniques for Training GANs". arXiv:1606.03498

[4] Isola, Phillip; Zhu, Jun-Yan; Zhou, Tinghui; Efros, Alexei (2017). "Image-to-Image Translation with Conditional Adversarial Nets". *Computer Vision and Pattern Recognition*.

[5] Ho, Jonathon; Ermon, Stefano (2016). "Generative Adversarial Imitation Learning". *Advances in Neural Information Processing Systems*: 4565–4573. arXiv:1606.03476. Bibcode:2016arXiv160603476H.

[6] “world’s largest doodling data set” https://quickdraw.withgoogle.com

[7] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434* (2015).

[8] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). *Generative Adversarial Networks* (PDF). Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

[9] generative adversarial networks with keras, https://github.com/coreyauger/quickdraw-gan

[10] Quickdraw-GAN, https://github.com/grantbey/quickdraw-GAN

[11] Wang, Chaoyue, et al. "Evolutionary generative adversarial networks." *IEEE Transactions on Evolutionary Computation* (2019).

[12] Manisha, Padala, and Sujit Gujar. "Generative Adversarial Networks (GANs): What it can generate and What it cannot?." *arXiv preprint arXiv:1804.00140* (2018).

[13] Jonathan Hui. "GAN — Why it is so hard to train Generative Adversarial Networks!" https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b (2018)

[14] Documentation on how to access and use the Quick, Draw! Dataset. https://quickdraw.withgoogle.com/data

[15] Lala, Sayeri, et al. "Evaluation of mode collapse in generative adversarial networks." *High Performance Extreme Computing, IEEE* (2018).