Learning by Competition: Dual Discriminator Generative Adversarial Networks

Carlos Andres Betancourt Baca, Wen-Hui Chen*, Wei-Hao Chen

Graduate Institute of Automation Technology
National Taipei University of Technology
Taipei, Taiwan

*whchen@ntut.edu.tw

Abstract

There has been a resurgence of interest on generative adversarial networks (GANs) in recent years. The overall performance of the generator depends on how well the discriminator is trained. In this study, we use two discriminators and one generator in the adversarial architecture. Each discriminator has a different perspective of how to evaluate the generated data, which makes the dual discriminators compete with each other to improve the performance of the generator output. The competition of the discriminators is evaluated using a zero-sum game in order to make the optimizer converge into the Nash-equilibrium. Experimental results show that the proposed approach has better visual performance and the EM-distance metric over some well-known GAN models, including DCGAN, BEGAN, WGAN, and WGAN-GP.

Keywords: generative models, dual discriminators, Nash-equilibrium, minimax optimization.

1. Introduction

GANs have many potential applications and been applied to various areas, which makes GANs an important research topic in deep learning. Successful applications include data augmentation [1], art generation [2], image-to-image translation [3]-[4], super-resolution [5], and image completion [6].

The idea behind GANs is to develop two neural networks, a generator and a discriminator, competing with each other [7]. The generator is trained to learn generating synthetic data that is likely to come from the sample space by taking a latent variable drawn from some distributions such as the normal distribution or uniform distribution. The

discriminator then evaluates the generated data and gives a scale value representing the probability that this generated sample comes from the real dataset. In other words, the discriminator is trained to distinguish the data generated by generator from the real data.

The optimization of a GAN network can be formulated as a minimax game. Given an optimal discriminator, the generator is learn to minimize the distribution difference between the generated data and the real sample data. Eventually, both the generator and the discriminator networks improve their performance, and in theory with a sufficient network capacity, the generator should be able to learn the underlying data distribution and generate samples that are impossible to be differentiated by an optimal discriminator.

However, the optimization of the traditional GAN algorithm sometimes tends to diverge as it is not easy to train and can have the mode collapse issue due to the mode seeking process where the model distribution concentrates on a single mode of data distribution [8]. A theoretical analysis made by [9] shows that training GANs tends to be unstable. Therefore, it is necessary to tune the network parameters carefully. Many improvements have been made to cope with these problems. Some of the theoretical breakthroughs can be found in [10]-[11].

2. Method description

In the original GAN paper, there are only two players for competition: a generator and a discriminator. Both the generator and the discriminator are parameterized by neural networks. As described above, the generator tries to counterfeit original data, for instance, a specific class of images from a particular dataset. The discriminator aims to

discern which samples are taken from the real dataset and which ones are fake. The generator only uses the information, a scalar value derived from the discriminator to improve its performance. The value given by the discriminator represents how close the discriminator consider the sample generated by the generator comes from the real dataset.

At the beginning, due to the random initialization of the networks parameters, the result could not be the best. After iteration runs for optimization of the objective function, the generator and the discriminator will update its parameters and improve their abilities. Researchers have shown that theoretically with a sufficient capacity of neural network and enough training [7], the samples created by the generator will look identical to those that come from the real data. No matter how good the discriminator is, it will have a 50% chance of correctly determining the class of both real and fake samples. This means that the generator learns the underlying distribution of the real dataset.

In the line of the thought described above, we use two discriminators to evaluate the performance of the generator to improve the performance of the generator. The authors in [12], also adopted multiple discriminators, but the difference between [12] and our approach is the way the networks are trained. The discriminators in [12] cooperate to get a better output and the generator receives information from multiple sources that make the generator improve its performance in a lower amount of epochs. The main issue of the approach in [12] is that it requires higher amount of memory to store the information of different discriminators, and the training-time per epoch increases, as multiple neural networks need to be trained. To cope with this issue, instead, we make the discriminators compete to each other to improve overall performance but use less memory.

The generator is a parametric function that maps a random variable z drawn from a fix probability distribution $p(z)$ corresponding element $x \in X$ that is in the same space of the real data. We define the generator as the function: $G_\theta(z)$ where θ represents the set of parameters of the mapping function, the weights and biases of a neural network. The discriminators similarly are defined as parametric functions $D_{\theta_i}(x)$, $i \in \{1, 2\}$, that assign a real value to every element of X representing its confidence about the origin of the sample x (i.e. real or generated). The generator competes against the two discriminators.

To train the generator, we maximize the expected value of $D_{\theta_1}(z) + D_{\theta_2}(z)$. To train the discriminators, we take a different approach by observing the symmetry of the

discriminators. We know that if we could train the dual discriminators to their optimality, the output of both discriminators should always be equal no matter their individual input.

We make the dual discriminators compete in a min-max game. Additionally, as the symmetry of the pay-off function is a zero-sum game, by applying the min-max theorem, we can guarantee that the optimal point will reach the Nash-equilibrium. This allows us to train one discriminator per epoch; thus, in order to keep the performance of both discriminators at similar levels, we only train the one with the lower performance. This avoids the need to optimize all of the neural networks in every epoch, reducing the execution time. The min-max game between the discriminators and the generator can be expressed in (1):

$$\min_G \max_D V(G, D_1, D_2) = \sum_{i=1}^2 \mathbb{E}_{x \sim p_{data}(x)} [D_{\theta_i}(x)] - \sum_{i=1}^2 \mathbb{E}_{z \sim p(z)} [D_{\theta_i}(G_\theta(z))] \quad (1)$$

To train the generator, we use a different approach. Notice that taking the gradients of (1) with respect to the parameters of D_{θ_i} , gives (2):

$$\nabla_{\theta_1} V = \nabla_{\theta_1} \mathbb{E}_{x \sim p_{data}(x)} [D_{\theta_1}(x)] - \nabla_{\theta_1} \mathbb{E}_{z \sim p(z)} [D_{\theta_1}(G_\theta(z))] \quad (2)$$

That is clearly independent from D_{θ_2} , similarly taking the partial derivatives with respect to the parameters of D_{θ_2} gives a symmetric expression that is independent of D_{θ_1} . If we use this rule to train the discriminators, these will not use the information of their performance related to each other. It is equivalent to use the training algorithm two times independently. In that case, we would not be able to take advantage of the symmetry of both discriminators. To avoid this, we define a different cost function to train the dual discriminators. First, we define the expected performance of the discriminators in (3):

$$E_i = f_k \left(\mathbb{E}_{x \sim p_{data}(x)} [D_{\theta_i}(x)] - \mathbb{E}_{z \sim p(z)} [D_{\theta_i}(G_\theta(z))] \right), \quad (3)$$

$$i \in \{1, 2\}$$

where $f_k(x)$ is the logistic function $1/(1 - e^{-kx})$

Hence, we can now define an appropriate cost function $C(E_1, E_2)$. Because of the symmetry of the discriminators, we use a zero-sum game to guarantee that the equilibrium point is the Nash-equilibrium. To have this, we need $C(E_1, E_2)$ to be anti-symmetric with respect to its parameters i.e. $C(a, b) = C(b, a)$. Note that this function

should not be a linear combination of E_1 and E_2 , because the gradients with respect to the parameters of one discriminator will be independent of the other. With the min-max theorem, we know that the Nash equilibrium of the game is given by the optimization defined in (4):

$$\nabla_{\theta_1} V = \nabla_{\theta_1} \mathbb{E}_{x \sim p_{data}(x)} [D_{\theta_1}(x)] - \nabla_{\theta_1} \mathbb{E}_{z \sim p(z)} [D_{\theta_1}(G_{\theta}(z))] \quad (4)$$

In order to avoid falling into the local optima in the cost function $C(a, b)$, we require it to be monotonically increasing with respect to the difference between a, b and satisfy (5). In this way, the cost will penalize more and more the competitor that is losing, being necessary to increase gradually the effort needed to surpass the leading one.

$$0 \leq a > b, \epsilon > 0 \Rightarrow C(a + \epsilon, b + \epsilon) > C(a, b) \quad (5)$$

The proposed cost function for the discriminators is:

$$C(a, b) = \text{atanh}(\eta(a - b)(1 + ab)) \quad (6)$$

where $\text{atanh}(x)$ is the inverse of the hyperbolic tangent function of x , and $a, b \in I, I = [0, 1]$. The limiting value $\eta \in (0, 1)$ guarantees that the cost function C is well-defined. Putting (3) in (6) gives:

$$C(E_1, E_2) = \text{atanh}(\eta(E_1 - E_2)(1 + E_1 E_2)) \quad (7)$$

If we train both discriminators at the same time, depending on the initial values, one of them should outperform the other. The algorithm only updates the discriminator with the lowest value; otherwise, one of the discriminators improves much faster than the other does, and the result is similar to only having one discriminator. In our experiment, the discriminators are trained more frequently than the generator. The procedure is summarized in algorithm 1.

3. Experimental Results

We trained several well-known models and compared their visual results with the results of our method using the LSUN-Bedroom dataset [13]. The LSUN-Bedroom dataset consists of 3,033,042 images that were center-cropped and resized to 64x64 pixels.

The trained models for comparison were: DCGAN [11], WGAN [10], WGAN-GP [14] and BEGAN [15]. The parameters remain the same as suggested in the references. The parameters used in our approach were: $n_{\text{disc}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0.5$, $\beta_2 = 0.9$, $k = 1000$. For all the models we

used a batch size of 64 samples, and ran for 250k iterations.

To quantitatively evaluate the performance of the generator, we choose the EM-distance as an evaluation metric [16]. The EM-distance or Wasserstein-1 is a measure between probability distributions and has some advantages over KL-divergence or JS-divergence. KL-divergence is well known for not being symmetric with respect to its parameters. Although JS-divergence is symmetric, i.e. $JS(P, Q) = JS(Q, P)$, it does not satisfy the triangular inequality and has discontinuity problems. The EM-distance is symmetric, satisfies the triangular inequality, and is continuous almost everywhere. To perform the computation of the experiment we used the open source code from [17].

According to our experiments, BEGAN obtained the worst results. After a point of iteration, the samples degenerated and failed to continue to improve, which is known as mode collapse. BEGAN is known for performing well on face datasets [15] such as CelebA [18]. This indicates that the tuning of the hyper-parameters is critical on different datasets.

DCGAN and WGAN performed better than BEGAN in our experiments. Figure 1 shows the results of the samples generated by DCGAN after the training. The results are clearly better than those generated by BEGAN in Fig. 2, but there are some collapsed samples. The results of WGAN are shown in Fig. 3. The performance of WGAN shows similar problems as those of DCGAN. Some samples are clearly unrealistic, but in general, the samples are comparable to those of DCGAN.

The samples generated by WGAN-GP are shown in Fig. 4. WGAN-GP and our method both show the best performance. Clearly, the shape of the figures is more realistic compared to those of DCGAN and WGAN. This indicates that the gradient penalty used in the algorithm instead of the clipping in WGAN makes the training more stable.

The samples generated by the proposed approach, shown in Fig. 5, are at least as good as those generated by WGAN-GP. The estimated EM-distance of our algorithm with respect to the real data is smaller than that of WGAN-GP, making our algorithm perform better on the LSUN dataset.

The EM-distance between the data generated by the generator and the real data is listed in Table. 1. The distance between BEGAN and the real data was not considered due to mode-collapse. The smaller the distance means that the distribution is closer to that of the original dataset.

Table 1. EM-distance between real and the generated data.

Algorithm	$EM(p_{data}(x), G_{\theta}(z))$	Mode collapse
DCGAN	4.71×10^{-3}	No
BEGAN	Mode collapse	Yes
WGAN	4.67×10^{-3}	No
WGAN-GP	4.55×10^{-3}	No
The proposed Approach	4.49×10^{-3}	No

Fig. 1. DCGAN, iteration: 250k.

Fig. 2. BEGAN, iteration: 150k.

Fig. 3. WGAN, iteration: 250k.

Fig. 4. WGAN-GP, iteration: 250k.

Fig. 5. Our method, iteration: 250k.

4. Conclusions

In this study, we have implemented a dual discriminator GAN network based on a competitive architecture. The visual results and the EM-distance of the experiment on the LSUN-Bedroom dataset showed that the proposed approach could obtain good results over some well-known methods. It indicates that using competing discriminators can lead to better performance than a traditional GAN framework. We are currently conducting the proposed approach on various datasets to verify the adaptability and scalability.

Algorithm 1 Competing GAN. Default values: $n_{disc} = 5, \alpha = 0.0001, \beta_1 = 0.5, \beta_2 = 0.9, k = 1000$

Require: The number of critic iterations per generator iteration n_{critic} , the batch size m , Adam hyper-parameters α , β_1 , and β_2 .

Require: Initial discriminators' parameters $\theta_{1,2}$, initial generator's parameters θ .

```

1: for number of training iterations do
2:   for  $n_{disc}$  steps do
3:     Sample real data minibatch
        $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\} \sim p_{data}(x)$ 
4:     Sample noise samples minibatch
        $\{z^{(1)}, z^{(2)}, \dots, z^{(m)}\} \sim p(z)$ 
5:      $\hat{x}^{(i)} \leftarrow G_{\theta}(z^{(i)})$ 
6:      $E_j \leftarrow f_k \left( \frac{1}{m} \sum_{i=1}^m (D_{\theta_j}(x^{(i)}) - D_{\theta_j}(\hat{x}^{(i)})) \right)$ 
7:      $L \leftarrow C(E_1, E_2)$ 
8:     if  $L \geq 0$  then
9:        $\theta_2 \leftarrow Adam(-\nabla_{\theta_2} L, \theta_2, \alpha, \beta_1, \beta_2)$ 
10:    else
11:       $\theta_1 \leftarrow Adam(\nabla_{\theta_1} L, \theta_1, \alpha, \beta_1, \beta_2)$ 
12:    end if
13:  end for
14:  Sample noise samples minibatch
      $\{z^{(1)}, z^{(2)}, \dots, z^{(m)}\} \sim p(z)$ 
15:   $\hat{x}^{(i)} \leftarrow G_{\theta}(z^{(i)})$ 
16:   $L_G \leftarrow \frac{1}{2m} \sum_{i=1}^m (D_{\theta_1}(\hat{x}^{(i)}) + D_{\theta_2}(\hat{x}^{(i)}))$ 
17:   $\theta \leftarrow Adam(-\nabla_{\theta} L_G, \theta, \alpha, \beta_1, \beta_2)$ 
18: end for

```

Acknowledgment

This work is supported by Taiwan Ministry of Science and Technology (MOST) under Grant No. MOST 106-2221-E-027-059

References

- (1) Antreas Antoniou, Amos Storkey, Harrison Edwards, "Data augmentation generative adversarial networks," March 2018, arXiv:1711.04340v3
- (2) Elgammal, Ahmed et al., "CAN: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms," June 2017, arXiv:1706.07068v1
- (3) Hao Dong et al., "Unsupervised image-to-image translation with generative adversarial networks," Jan. 2017, arXiv:1701.02676v1
- (4) Ming-Yu Liu, Thomas Breuel, and Jan Kautz, "Unsupervised image-to-image translation networks," July, 2018, arXiv:1703.00848v6

- (5) Jun-Yan Zhu et al., "Unpaired image-to-image translation using cycle-consistent adversarial networks," Feb. 2017, arXiv:1703.10593v4
- (6) Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 107:1-14, July 2017.
- (7) Ian Goodfellow et al., "Generative adversarial networks," June 2014, arXiv:1406.2661v1
- (8) Tu Dinh Nguyen et al., "Dual discriminator generative adversarial nets," September 2017, arXiv:1709.03831v1
- (9) Martin Arjovsky and Léon Bottou. "Towards principled methods for training generative adversarial networks," Jan. 2017, arXiv:1701.04862v1
- (10) Martin Arjovsky, Soumith Chintala, Léon Bottou, "Wasserstein GAN," Dec. 2017, arXiv:1701.07875v3
- (11) Alec Radford, Luke Metz, Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," Jan. 2016, arXiv:1511.06434v2
- (12) Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan, "Generative multi-adversarial networks," March 2017, arXiv:1611.01673v3
- (13) Fisher Yu et al., "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," June 2016, arXiv:1506.03365v3
- (14) Ishaan Gulrajani et al., "Improved training of Wasserstein GANs," Dec. 2017, arXiv:1704.00028v3
- (15) David Berthelot, Thomas Schumm, Luke Metz, "BEGAN: boundary equilibrium generative adversarial networks," May 2017, arXiv:1703.10717v4
- (16) Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99-121, Nov. 2000.
- (17) Gary Doran, "PyEMD: Earth mover's distance for Python," 2014, <https://github.com/garydoranjr/pyemd>
- (18) Ziwei Liu et al., "Deep learning face attributes in the wild," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730-3738, 2015.