University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 21: November 6th

Lecturer: Alexander Schwing Scribe: Chenzhang Xiao

1 Overview

This lecture [6] introduces a novel generative machine learning algorithm known as generative adversarial nets (GAN), which is proposed by Goodfellow et al. in 2014 [4]. It utilizes two competing deep neural networks, a discriminator and a generator, to directly create data samples matching the distribution of the training data. In this algorithm, the generator creates artificial data samples, while the discriminator tries to determine whether an input data sample is from the original training data set or from the generator. As two neural nets compete against each other, they are both improved until the discriminator can no longer distinguish artificial data from the training data, meaning the distribution of artificial samples are perfectly matching that of the original data set.

In this lecture, we are going to formulate this two-player game mathematically and analyze global optimality of this formulation assuming unlimited capacity of the discriminator and the generator. Implementation of this algorithm will also be introduced, followed by a few experimental results. An alternative formulation to traditional GAN, known as Wessterian GAN (WGAN), is also introduced, which utilizes a differentiable measure of distance between two distributions to train the model and thus increase the stability of the algorithm and prevent mode collapse.

In general, goals of this lecture include

- Getting to know Generative Adversarial Nets
- Understanding generative methods
- Differentiating between discriminative and generative methods

2 Background Information

2.1 Discriminative models

So far in the class we have learned a few supervised machine learning algorithms such as logistic regression, SVM, deep nets, etc. In these methods, we train the model with data set x and previously assigned labels y, and obtain the optimal model parameters w by maximization of the likelihood of agreement between assigned labels and model predicted labels.

$$p(y|x) = \frac{exp(F(x, y, w))/\epsilon}{\sum_{\hat{y}} exp(F(x, \hat{y}, w))/\epsilon} \tag{1}$$

In general, these models try to predict the distribution of y given data x, we call them discriminative models.

2.2 Generative models

A few unsupervised machine learning algorithms have also been introduced: k-means, Gaussian mixture models (GMM), hidden Markov models (HMM) and variational auto-encoder(VAE). In

these models, labels of data are unavailable before training the model, and we obtain the model parameters by maximization of the likelihood of matching of data distribution and generated model distribution.

$$\theta^* = \max_\theta \sum_i \log p(x^{(i)}; \theta) \tag{2}$$

In general, these models try to predict the distribution of data x, $p(x)$, and we refer to these models as generative models.

2.3 Why studying generative models

Goodfellow pointed a few reasons why generative model are worth studying in his tutorial of GAN in 2016 [3]:

- To test our ability to represent and manipulate high-dimensional probability distribution, such as images repenting a group of objects
- To incorporate these models in reinforcement learning
- To provide prediction on inputs that have missing data

2.4 How to model data distribution

Adopted by almost all traditional generative methods, one way to model data distribution is to obtain an explicit function of data distribution with exact model parameters. However, to obtain valid model parameters that matches data distribution, people have to introduce noise terms to the model distribution to cover all the examples. As a a result, in applications such as image generation, introducing noise terms would reduce the quality of outcome samples and make them very blurry, even though these noise terms are essential to maximize the likelihood of agreement between two distributions. In some other cases, the model parameters themselves could be difficult to find due to high dimension of the sample.

Instead of finding parameters of a probabilistic density function, people have turned to other approaches to create algorithms that indirectly model the distribution and sample data and generate samples with a similar distribution of the training data by finding a distribution in a lower-dimension manifold. Variational auto-encoder from the last lecture [6] and GAN from this lecture are two models adopting this approach.

3 GAN: Problem Formulation

As mentioned previously, the new frame work of generative adversarial nets consists of two competing networks, a discriminator and a generator. For the generator to learn the distribution over data x, a prior on input noise variable $p_z(z)$ and a mapping function to data space x as G_θ, where θ represents model parameters of the generator, will be determined. For the discriminator, we also define a model $D_w(x)$ with parameters w and determines the probability that x is from the original data rather than the generated data.

We will train D_w to maximize the probability of assigning correct labels to corresponding data sample (or minimize the negative log likelihood) and simultaneously train G_θ to minimize the probability of D_w assigning correct label to the generated data (G_θ). In other words, we can formulate this problem as a max-min game between two players G and D with the following value function V.

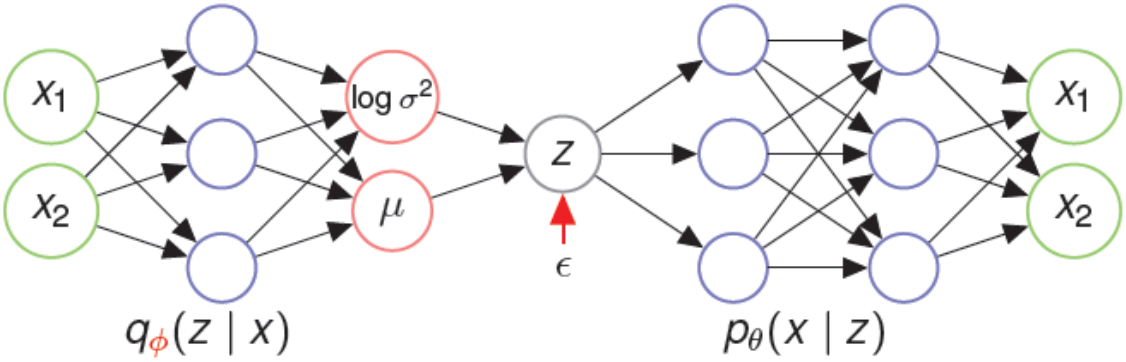

Figure 1: In VAE, two networks, an encoder and a decoder, are paired together. The encoder takes training sample as input and outputs the distribution parameters in a low-dimension space Z. Then we train another deep neural network and decode samples in Z to recover training samples. In this case, we can directly sample from Z space to obtain data with similar distribution as our original training data.

$$\max_{\theta} \min_{w} V(G, D) = -\sum_{x} log D_w(x) - \sum_{z} log(1 - D_w(G_\theta(z))) \tag{3}$$

To solve this problem, we will alternatively optimizing D and G until a set criteria is met. This process is presented in Algorithm 1. Graphical representations are also presented in Fig. 2 and Fig. 3 [4].

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Moreover, when we have a good discriminator and bad generator, there is almost no gradient for the generator G to change. In this case, instead of solving the above equation, we then to use $\min_\theta -\sum_z logD_w(G_\theta(z))$(Fig.4). However, the joint distribution is D and G is ignored using this equation.

4 Theoretical Analysis

In this section, we would like to present a theoretical analysis of GAN when D and G are given unlimited capacity. We will show that the proposed max-min game of $V(G, D)$ has a global optimum when $p_{data} = p_g$.

Let's first consider the optimal discriminator D^* when given an arbitrary generator G. In this case, we should find a D that minimize the value function $V(G, D)$.

$$\begin{aligned}\min_D V(G,D) &= -\int_x p_{data}(x)logD(x)dx - \int_z p_z(z)log(1-G_\theta(z))dz \\ &= -\int_x p_{data}(x)logD(x) - p_G(x)log(1-D(x))dx\end{aligned} \tag{4}$$

Now we formulate the equation above to be Eq. 5 and use Eluer-Lagrange equation to obtain Eq. 6.

$$V(D) = \int_x L(x, D, \dot{D}dx) \tag{5}$$

$$\frac{\partial L(x,D,\dot{D})}{\partial D} = \frac{d}{dx}\frac{\partial L(x,D,\dot{(D)})}{\partial \dot{D}} \tag{6}$$

Plug in expression for L, we can see the right side of Eq. 6 equals to 0 and thus we can directly solve for its left side (Eq.7) and consequentially obtain the optimal D^* that minimizes this value function (Eq.8).

$$\frac{\partial L(x,D,\dot{D})}{\partial D} = -\frac{p_{data}}{D} + \frac{p_G}{1-G} = 0 \tag{7}$$

$$D^* = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \tag{8}$$

Given the optimal discriminator, we can rewrite the value function V and reformulate this max-min game as:

$$\begin{aligned}C(G) = \min_D V(G,D) &= -\int_x p_{data}(x)logD^*(x) + p_G(x)log(1-D^*(x))dx \\ &= -\int_x p_{data}(x)log(\frac{p_{data}(x)}{p_{data}(x)+p_G(x)}) + p_G(x)log(\frac{p_G(x)}{p_{data}(x)+p_G(x)})dx\end{aligned} \tag{9}$$

We recognize that we can easily rewrite Eq.9 using Kullback-Leibler divergence and further more using Jensen-Shannon divergence, a measure of similarity between two probability distributions, as shown in Eq.10 and Eq.11.

$$C(G) = log(4) - KL(p_{data}, \frac{p_{data}(x) + p_G(x)}{2}) + KL(p_G, \frac{p_{data}(x) + p_G(x)}{2}) \tag{10}$$

$$C(G) = log(4) - 2 \cdot JSD(p_{data}, p_G) \tag{11}$$

Eq.11 has a global maximal value of $log(4)$ when $p_{data} = p_G$, as Jensen-Shannon divergence is always non-negative, and equals to zero iff two probabilistic distributions are equal. In our case, the generative model perfectly matches with the model of original data when we reach this global optimal point. Hence, we have proved that this max-min game has a single global optimal point when $p_{data} = p_G$.

5 Alternatives

After publication of GAN in 2014, researches on GAN increases exponentially. A few modifications and alternations of GAN, such as deep convolutional GAN (DCGAN) [5], StackGAN [7], Wasserstein GAN (WGAN) [1] are proposed and shown to exhibit better convergence rate or improved stability compared with traditional GAN.

5.1 WGAN

WGAN utilizes the Earth-Mover(EM) distance, or Wassersterin-1 (Eq.12) instead of the JS divergence term used in traditional GAN.

$$W(p_{data}, p_G) = \min_{p_J(x,x') \in \Pi(p_{data}, p_g)} \mathbb{E}_{p_J}[\|x - x'\|] \tag{12}$$

Intuitively, the EM distance indicates how much "mass" should be removed from x to x' in order to transform the distribution p_G to p_{data}. Such measure of difference between two probabilistic distributions has been proved to be continuous and provide a useful gradient that we can utilized to learn a desired distribution, while other measures of divergence or difference such as total variation (TV), KL and JS are not even continuous and don't have such a nice property.

In this case, we can formulate the original max-min problem as:

$$\min_{P_G} W(p_{data}, p_G) = \min_{p_G} \min_{p_J(x,x) \in \Pi(p_{data}, p_g)} \mathbb{E}_{p_J}[\|x - x'\|] \tag{13}$$

Since p_{data} is not directly available, we can further use Kantorovich-Rubinstein duality and rewrite $W(p_{data}, p_G)$ as:

$$W(p_{data}, p_G) = \max_{\|f\|_L \leq 1} \mathbb{E}_{p_{data}}[f(x)] - \mathbb{E}_{p_G}[f(x')] \tag{14}$$

Now we have the min-max problem as

$$\min_{p_G} \max_{w} \mathbb{E}_{p_{data}}[f_w(x)] - \mathbb{E}_{p_G}[f_w(x')] \tag{15}$$

where $w \in W$ represents a set of parameters of function f, which could be a neural network. further detail of WGAN can be found in .

5.2 Benefits of WGAN

WGAN provides two benefits compared with traditional GAN: 1) it provides a meaning loss function that correlates the convergence of the model distribution thus provide intuitive information during training, 2) it shows better stability compared with traditional GAN. From empirical results, no mode collapse is observed during training of WGAN.

EM distance a continuous and differentiable measure of "distance" between two probabilistic distribution, which means that we can easily utilize its gradient for training. In addition, as we approach optimality, the gradient of JS divergence gets closer to 0, while Wasserstein is always also reliable (Fig.5 [1]). A proof of concept is also illustrated in Fig.6 [1]. In addition, due to the ability to reach optimality using this loss function, mode collapse is not observed if the networks are properly set up.

6 Results

In this section we presented a few results of image generation using several GANs.

6.1 Traditional GAN

GAN is trained over a range of data sets including MNIST, teh Tronote Face Database, and CIFAR-10 (Fig. 7, [4]). The model has not memory of the training set and generated images looks very sharp.

6.2 DCGAN

DCGAN is proposed by Radford et al.in 2015 [5], it utilizes convolutional neural networks (CNN) for the discriminator and generator. Their reported results are shown in Fig.8.

6.3 StackGAN

Zhang et al. [7] proposed a Stacked Generative Adversarial Networks (StackGAN) to generate 256x256 photo-realistic images conditioned on text descriptions. A two-stage method is used such that the Stage-I GAN creates a low-resolution image with primitive shape and colors of the object based on the given text description, and a Stage-II GAN is further utilized to generate a higher resolution image with realistic features based on results from Stage-I GAN. Their reported results are shown in Figure.9 [7].

6.4 WGAN

Images generated using WGAN formulation and traditional GAN formulation are compared in Fig.10 [1]. WGAN shows similar image quality and better stability compared with traditional GAN formulation.

7 Advantage and Disadvantage

This new frame work of GAN has several advantages and disadvantages compared with previous generative models. One foremost advantage of GAN is its ability to generate sharp and realistic

image. Due to introduction of noise and bias in other generative machine learning algorithms, the generated images are often blurry. This is why GAN is so fascinating in the first place. Goodfellow also argued that GAN generates samples faster compared with other generative models, does not need Monta Carle approximation for training, which is used in Boltzmen machine, and thus easier to train [2]. However, GAN is known to be very unstable. The convergence of GAN is theoretically guaranteed given unlimited capacity of the discriminator and generator while in reality such limitation exists and the final Nash equilibrium is sometimes hard to achieve. GAN is also known for other issues such as mode collapse, oscillation and not able of dealing with discrete sample such as text. In addition, GAN does not provide an explicit distribution function of sample data, which is required in some applications.

8 Conclusion and Discussion

In this lecture, the difference between discriminative models are generative models are presented. Even though they all used the same concept of likelihood maximization methodology, the fundamental purpose is essentially different. As discriminative models try to predict the conditional probability of label y given data x, generative models is used to model the probabilistic distribution of x.

While traditional generative algorithms try to determine model parameters explicitly, a few other methods choose a different path to directly generate sample data instead, such as GAN introduced in this lecture. In some applications, the ability to generate samples is more important than knowing the explicit model distribution function. In addition, due to the complicated distribution in higher dimensional examples such as images, finding the explicit expression will result in the introduction of noise and bias to cover all different samples, thus makes generated images blurry. On the other hand, GAN is capable of generating realistic images due to its nature of embedding model distribution inside a trained neural network.

GAN utilized two competing neural networks, a generator creating artificial samples to match real samples, and a discriminator tries to determine if the sample is artificial or real. Such competition results in a Nash equilibrium where the discriminator is no longer capable of determine whether a sample is from the generator or from the training data. We formulate this process into max-min mathematical game and utilize JS divergence to calculate difference between model distribution and data distribution to update two networks. We have also proved that given unlimited capacity of G and D, we can reach a global optimal point where model distribution equals to the data distribution.

In spite of its ability to generate realistic images, GAN is known to be unstable and hard to train. Researcher have proposed some modifications or alternative formulations of GAN. WGAN is one of the alternative formulations proposed. It utilizes EM mover to measure the difference between two distributions instead of JS divergence used in traditional GAN and provides a more stable training process.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[2] I. Goodfellow. Qura what are the pros and cons of using generative adversarial networks (a type of neural network)? could they be applied to things like audio waveform via rnn? why or why not? https://www.quora.com/What-are-the-pros-and-cons-of-using-generative-adversarial-networks-a-type-of-neural-network-Could-they-be-applied-to-things-like-audio-waveform-via-RNN-Why-or-why-not. Accessed: 2018-11-19.

[3] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

[6] A. Schwing. Lecture notes in *ece*544*na* patter recognition, September 2018.

[7] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint*, 2017.

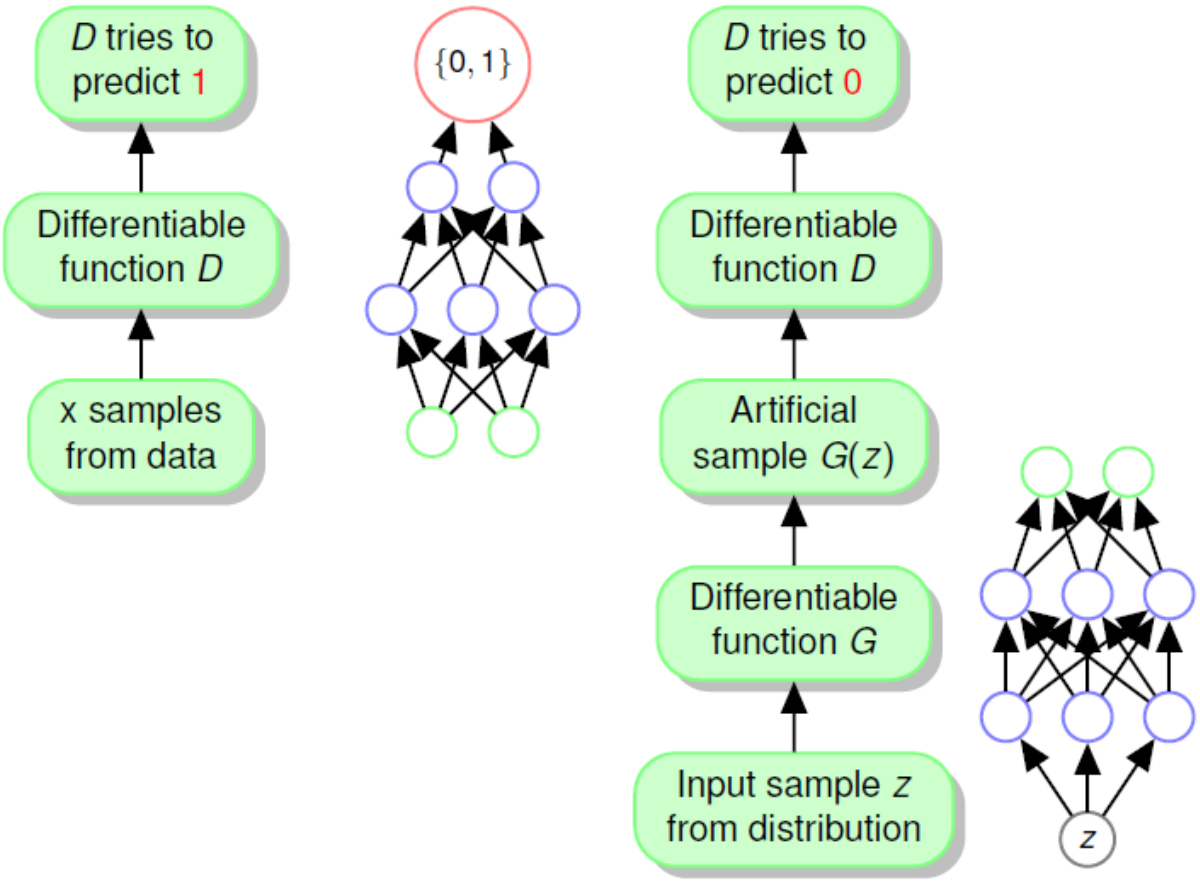

Figure 2: The GAN framework contains two players G and D, each presented by a differentiable function control by a set of parameters θ and w, against each other in a game. These function are usually implemented ad deep neural nets (DNN). The game goes in the following two scenarios. In the first scenario, the training sample x is randomly selective from the training set and used as the input to the discriminator, D, whose goal is to accurately determine real sample and artificial sample. In this case, it drives $D(x)$ to be as close to 1 as possible. In the second scenario, a random noise input z is used as the input to the second player, the generator. The output of this function, $G(z)$, is received by the discriminator. The discriminator tries to make $D(G(z))$ to be as close to 0 as possible while the generator strives to make it 1. In this case, if both player has the same capacity, $D(x) = \frac{1}{2}$ at the end of game, which means the training data and generated data have the same distribution.

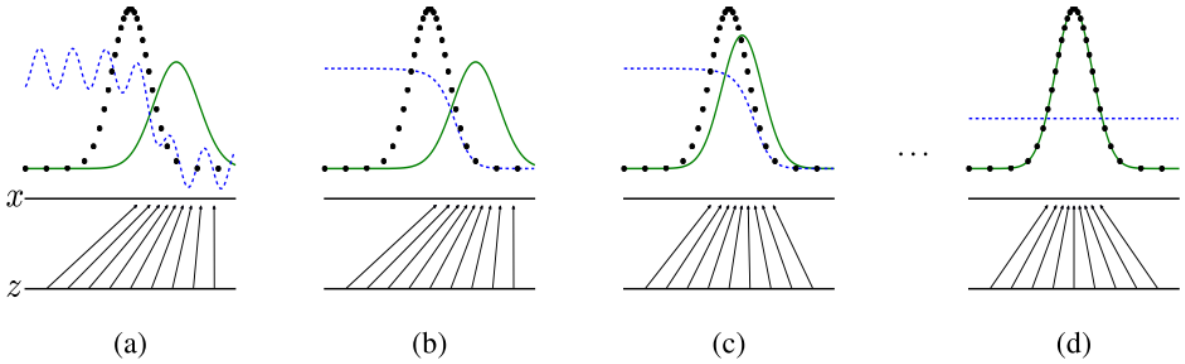

Figure 3: Another pictorial representation of GAN. Two networks are simultaneously updated during the training process as the discriminator tries to determine samples from training data (p_{data} given by the black dotted line) and generated data (p_g given by the green solid line. The upward arrows shows the mapping $x = G(z)$). a) at the very beginning p_{data} is similar to p_g and the discriminator is partially correct. b) the discriminator is better trained to distinguish two types of data. c) G is updated using the gradient of D such that $G(z)$ goes into regions that are more likely to be classified as training data. d) after many iterations the discriminator is no longer able to distinguish p_{data} and p_g and thus $D(x) = 1/2$.

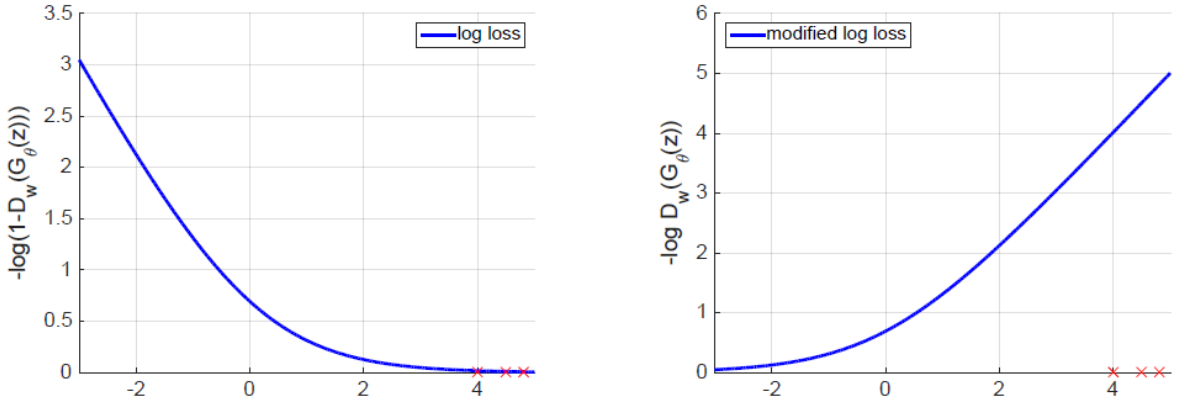

Figure 4: Figure on the left shows the log loss function in the value function. If we start somewhere on the left of the curve, the gradient is very slow and hard for it to move to the left. In addition, as we are trying to maximize this log function, there is no upper bound and could cause some issues in practice. In this case, a modified log function as shown in right figure is used. Instead of maximize the original log loss function, now we are minimizing this modified function. As we can see in the figure, the gradient is significantly higher and this function has a lower bound, which is more useful in practice.

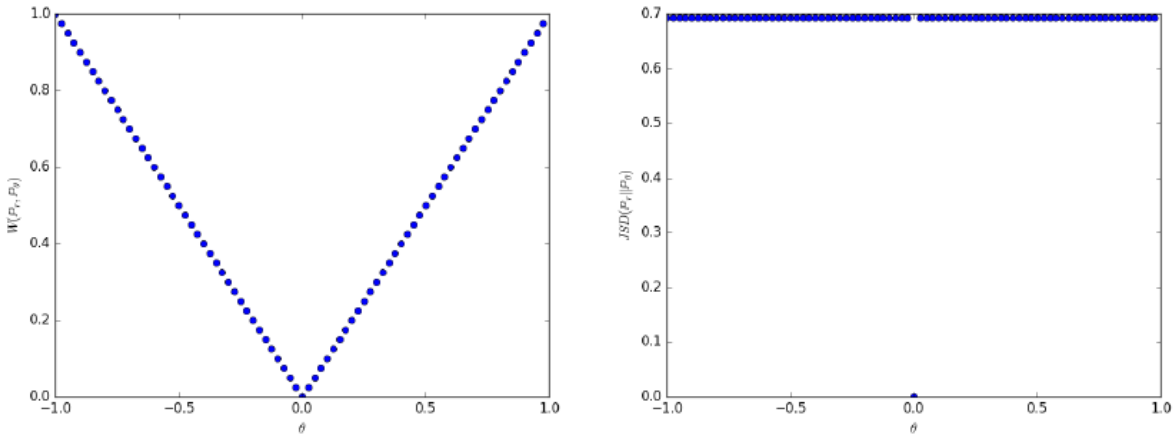

Figure 5: These plots shows $\rho(P_{G_\theta}, P_0)$ as a function of θ where ρ is the EM distance (left plot) and the JS divergence (right plot). We can clearly see that the left plot is piece-wise continuous and provides a useful gradient everywhere.

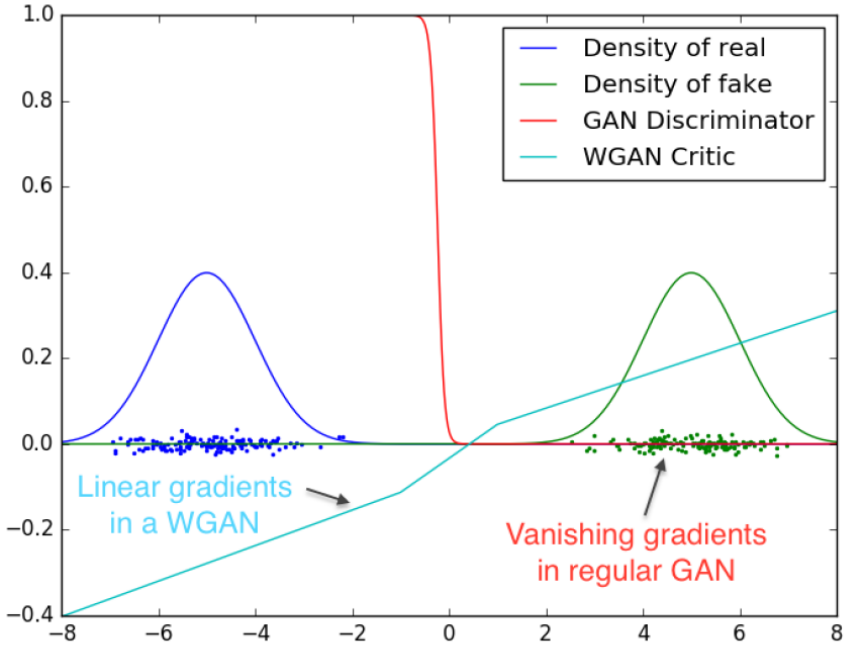

Figure 6: The discriminator of traditional GAN results in a vanishing gradient, while the criteria of WGAN provides a clean gradient in all spaces

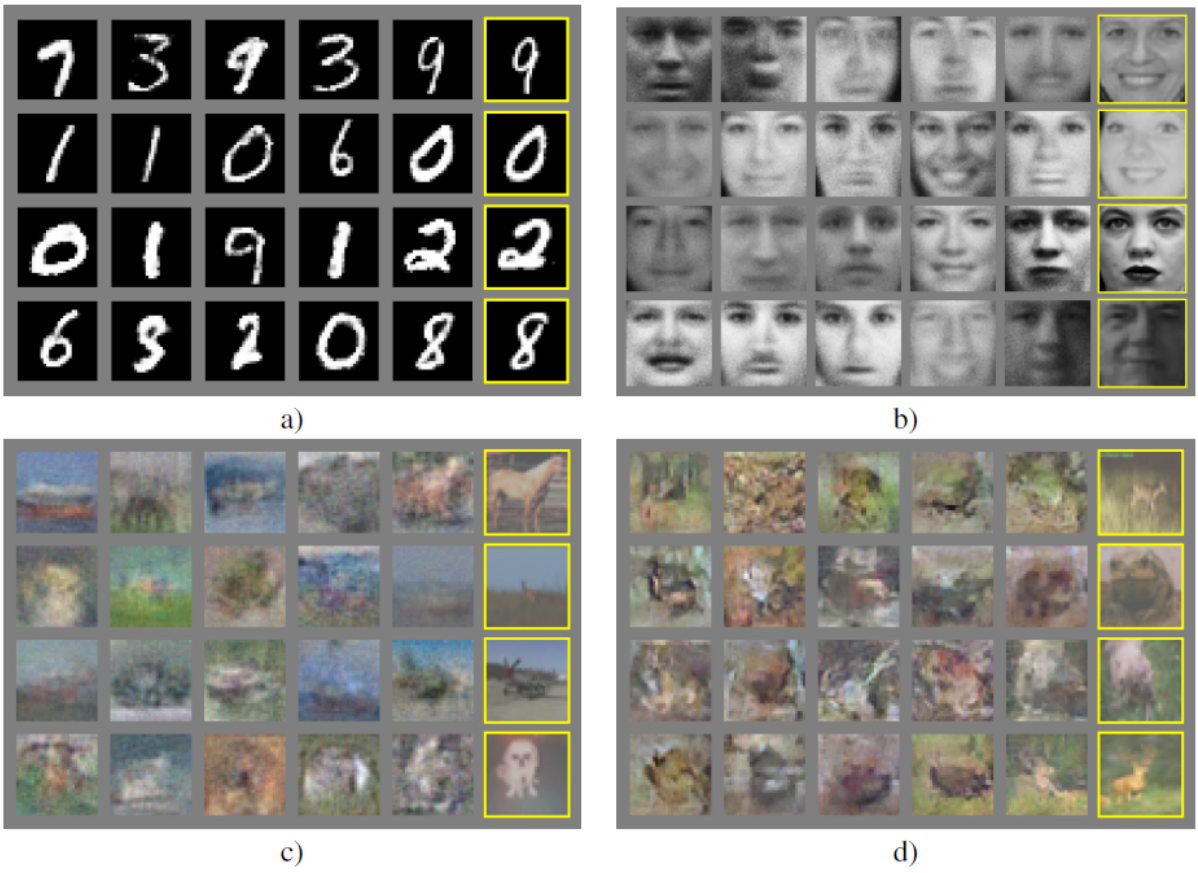

Figure 7: Samples generation from GAN. Rightmost colums shows the nearest training example of the neighboring sample.

Figure 8: Samples generated using DCGAN presents even more realistic features.

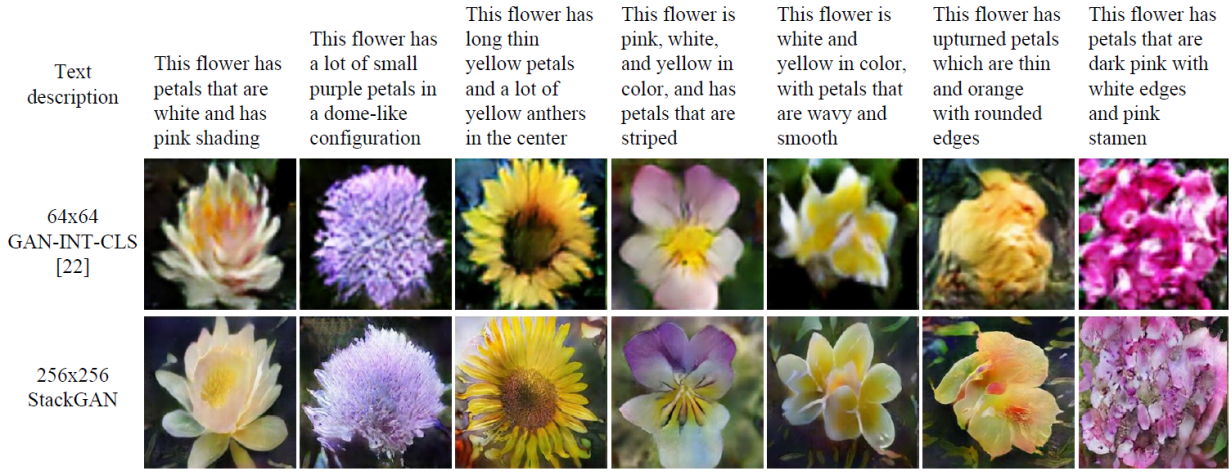

Figure 9: Images generated from text description using StackGAN from Oxford-102 set.

Figure 10: Top image compares results generated with a DCGAN generator. The top left images are from WGAN and top right images are from standard GAN formulation. Both algorithms are able to generate high quality samples. Lower images shows results from an algorithm trained without batch normalization and constant number of filters at every layer. The lower left images are from WGAN while the lower right images are from standard GAN formulation. As we can see WGAN is more stable as GAN failed to generated meaningful samples in this case.