Investigation on GAN, WGAN and WGAN-GP

Xiaozhu Fang
Department of EECS
University of Michigan
Ann Arbor, MI 48109
`fangxz@umich.edu`

Mingjie Gao
Department of EECS
University of Michigan
Ann Arbor, MI 48109
`gmingjie@umich.edu`

Shuyang Huang
Department of EECS
University of Michigan
Ann Arbor, MI 48109
`shuyangh@umich.edu`

Jiaren Zou
Department of EECS
University of Michigan
Ann Arbor, MI 48109
`jiarenz@umich.edu`

Abstract

In this project we implemented a Generative Adversarial Network (GAN), a Wasserstein Generative Adversarial Network (WGAN) and a WGAN with gradient penalty (WGAN-GP). GAN is famous for producing synthetic images from a learned distribution by minimizing the Jensen-Shannon divergence (JSD), whereas it lacks training stability. WGAN stabilizes the training by minimizing the earth mover's distance distance (EMD) instead of the JSD. WGAN-GP takes one step further by adding the gradient penalty to the EMD estimate. We studied the theoretical background behind these generative networks and understood the keys and advantages of each network by illustrative simulations. Then we applied these networks to the Carvana Image Masking Challenge dataset. The result matched our expectation: WGAN-GP outperforms others regardless of the architecture. GAN had the issue of instability and mode collapse in the training.

1 Introduction

Generative models, an interesting topic within machine learning field for years, can produce massive synthetic images of specific groups. The produced images are hardly discriminated by human. A striking success in this field was made by Generative Adversarial Networks (GAN) [1]. It simultaneously trains a generator to counterfeit fake images by inferring the distribution and a discriminator to distinguish the fake images from the generator and real images. Based on the information theory, the adversarial training is based on the Jensen-Shannon divergence between fake and real distributions. GAN successfully generated visually appealing images [2][3]. Apart from the images production, GAN achieved state-of-the-art performances for image super-resolution tasks [4][5], image inpainting [6], and undersampled magnetic resonance image (MRI) reconstruction [7]. However, it suffers from unstable training and is hard to use in practice. Stabilizing GAN is not trivial and many efforts have been devoted into this task [8][9].

Arjovsky et. al. [10] tackled the problem of training stability by proposing a new training scheme, and named it Wasserstein Generative Adversarial Networks (WGAN). WGAN minimizes the earth mover's distance (EMD) instead of the Jensen-Shannon divergence (JSD) between two distributions. Arjovsky showed by theory that EMD has more favorable properties and demonstrated by empirical results that WGAN enjoys a more stable training.

WGAN requires the discriminators to be Lipschitz functions. In the original WGAN paper, the Lipschitz constraint is imposed by weight clipping, which can lead to undesirable behaviors. Gulrajani added a penalty term to the loss function to penalize the gradient of the discriminators, and named it WGAN with gradient penalty (WGAN-GP) [11]. WGAN-GP showed a more stable training and a better image synthesis quality than WGAN.

In this project we implemented GAN, WGAN and WGAN-GP. Our contributions includes two aspects. On one hand, we reiterate the theory of these generative networks and explain their advantages. To illustrate that, we provide the straightforward simulations. On the other hand, we apply these GAN algorithms to the Carvana Image Masking Challenge dataset from *Kaggle*, showing the consistency of result with our understanding.

2 Related work

Since conventional GANs subject to several drawbacks including mode dropping phenomenon [12], absence of correlation between generated sample quality and training loss [10], and requirement of careful network and training scheme designs, a number of efforts have been made towards stable training of GANs [13, 14]. Novel training techniques have been proposed for this purpose. Several techniques such as feature matching, minibatch feature, and virtual batch normalization encouraging more stable training of GANs were proposed [8], by which semi-supervised learning performance and sample generation were improved. Instance noise [5] was another trick that can be applied to address instability problems. Metz introduced an unrolled optimization of the discriminator which enables switching between the ideal discriminator and the current discriminator during training to allow suppression of common problems in GAN training, such as mode collapse while increase the diversity of generator data distribution [9].

Another direction towards stabilizing GAN training is optimizing the objective function. In particular, Arjovsky theoretically analyzed the training dynamics of GANs and explored the source of the training instability of GANs [12]. For image super-resolution application, Sonderby presented a GAN variant aimed to minimize the KL divergence and cross-entropy between the high resolution data distribution and the model's output distribution [5]. Any f-divergence was proven valid for generative neural sampler training [15], and various divergence functions were compared with respect to training complexity and generator quality. Poole derived a family of generator objectives which correspond to arbitrary f-divergences, which improved sample quality or sample diversity [14]. WGAN which leverages Wasserstein distance to define an objective function which is theoretically superior than the previously proposed ones for its continuity and weaker topology. WGAN drastically reduces the mode dropping phenomenon and provides meaningful training curves which highly correlate with sample quality and useful for hyperparameter tuning.

3 Methods

3.1 Generative Adversarial Networks (GAN)

As discussed in EECS 545 class, a generative model is based on a full probabilistic assumption of the data. Under an unsupervised learning setting, it tries to learn a probability distribution $p(x)$ of the training data. The seminal work of Goodfellow et. al. proposes to train an adversarial net to obtain the generative model [1]. The adversarial training process simultaneously trains a generator G to approximate the data distribution and a discriminator D to determine whether a sample comes from the generator or real data. Mathematically, the process can be expressed as a minimax optimization problem

$$\min_G \max_D \mathbb{E}_{x\sim\mathbb{P}_r}[\log D(x)] + \mathbb{E}_{z\sim\mathbb{P}_z}[\log(1 - D(G(z))], \tag{1}$$

where $\mathbb{P}_r$ is the distribution of the real images and is what we want our generative model G to learn. Because G is an "implicit model", we cannot calculate the exact probability of each point. Instead, we can only sample the probability distribution and get images from it. $z \sim \mathbb{P}_z$ is a vector following a simple noise distribution to serve as the sample point of G. We denote the distribution of $G(z)$ as $\mathbb{P}_g$.

The $\max$ term has an interpretation of Jensen-Shannon (JS) divergence. We define JS divergence as

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r||\mathbb{P}_m) + KL(\mathbb{P}_g||\mathbb{P}_m), \tag{2}$$

where

$$KL(\mathbb{P}_r||\mathbb{P}_g) = \mathbb{E}_{x\sim\mathbb{P}_r}[\log(\frac{\mathbb{P}_r(x)}{\mathbb{P}_g(x)})] = \int \log(\frac{\mathbb{P}_r(x)}{\mathbb{P}_g(x)})\mathbb{P}_r(x)d\mu(x) \tag{3}$$

is the Kullback–Leibler (KL) divergence of the two distributions, as we learned in EECS 545 class, and $\mathbb{P}_m = (\mathbb{P}_r + \mathbb{P}_g)/2$. KL divergence is a very common metric to measure two probability, and it is widely used in Variational Inference and Variational Auto encoder.

From **Proposition 1** in [1], for a fixed G the optimal discriminator D^* is

$$D^* = \arg\max_D \mathbb{E}_{x\sim\mathbb{P}_r}[\log D(x)] + \mathbb{E}_{z\sim\mathbb{P}_z}[\log(1 - D(G(z)))] = \frac{\mathbb{P}_r}{\mathbb{P}_g + \mathbb{P}_r}. \tag{4}$$

Then Eqn. 1 becomes

$$\min_G \mathbb{E}_{x\sim\mathbb{P}_r}[\log D^*(x)] + \mathbb{E}_{x\sim\mathbb{P}_g}[\log(1 - D^*(x))] \tag{5}$$

$$= \min_G \mathbb{E}_{x\sim\mathbb{P}_r}[\log \frac{\mathbb{P}_r}{\mathbb{P}_g + \mathbb{P}_r}] + \mathbb{E}_{x\sim\mathbb{P}_g}[\log \frac{\mathbb{P}_g}{\mathbb{P}_g + \mathbb{P}_r}] \tag{6}$$

$$= \min_G JS(\mathbb{P}_r, \mathbb{P}_g) - \log 4. \tag{7}$$

To summarize, in GAN the training leads to the minimization of the JS divergence between the generated data distribution and the real data distribution.

3.2 Wasserstein Generative Adversarial Networks (WGAN)

As Goodfellow addressed, GAN has the issue of instability that deserves further investigation. Basically, the problem arises from the loss function, JS divergence. JS divergence is the most popular metric but not the unique one. An alternative metric of loss is the Wasserstein distance, also called earth mover's distance (EMD),

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma\in(\mathbb{P}_r,\mathbb{P}_g)} \mathbb{E}_{(x,y)\ \gamma}[||x - y||] \tag{8}$$

Superiority of EMD EMD is a metric to measure the distance between two distributions. It has the same functionality of JS divergence. To illustrate the benefits of EMD over JS divergence, we provide a simple simulation.

Assume we have two different Gaussian distributions in one dimension. Let $\mathbb{X}_1 \sim \mathcal{N}(0, C)$ and $X_2 \sim \mathcal{N}(d, C)$, where d denotes the distance between the means and they have the same variance C. We show the value of $JS(\mathbb{X}_1, \mathbb{X}_2)$ and $W(\mathbb{X}_1, \mathbb{X}_2)$ for three cases when $C = 0.1, 0.3, 1$.

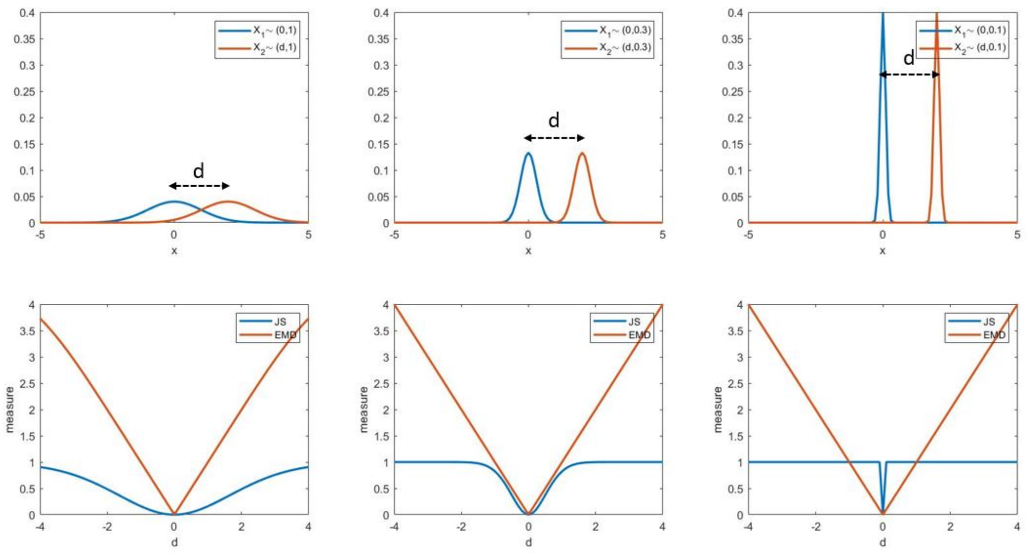


Figure 1: $JS(\mathbb{X}_1, \mathbb{X}_2)$ and $W(\mathbb{X}_1, \mathbb{X}_2)$(EMD) for two Gaussian distributions. The gradient of JS vanishes in contrast with invariant EMD, which shows the advantage of EMD.The detailed description of how we generated the figure is in Appendix.

According to Fig. 1, we can conclude: 1) Both JS and EMD are symmetric, and they achieve the minimum 0 at $d = 0$. The existence of minimum 0 indicates they share the same distribution; 2) When variance C increases, EMD keeps the steep gradient while the gradient of JSD vanishes. It accounts for the instability issue of JS when two distribution are not overlapped.

Discriminator in WGAN The job of discriminator is to minimize EMD of two distributions as Eqn. 9. Let γ be the joint distribution of $\mathbb{P}_r$ and $\mathbb{P}_g$. Inversely, $\mathbb{P}_r$ and $\mathbb{P}_g$ become the marginal of γ. Thus, γ is not uniquely determined with respect to its marginals. EMD is the solution of γ minimizing the expectation $\mathbb{E}[||x - y||]$. If all the distributions are discrete with the limited support, $\mathbb{W}(\mathbb{P}_r, \mathbb{P}_g) = \gamma^*$ of Eqn. 8 can be analytically solved by linear programming (LP) because the marginal distribution is exactly the constraints. Suppose X and Y both have **m** states, the linear programming would deal with 2**m** constraints. As **m** turns larger, however, LP associated with tremendous constraints eventually becomes intractable. This is the primary challenge of EMD and we resort to the mathematicians.

Actually EMD, referring to its name, is a transportation problem. Soviet mathematician Kantorovich [16] came up with a duality, which is applied on Eqn. 8.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in (\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\ \gamma}[||x - y||] = \max_{||f||_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_g}[f(x)] \tag{9}$$

where $f(x)$ is 1-Lipschitz continuous function. Furthermore, it also works for K-Lipschitz continuous if we scale W by $\frac{1}{K}$. Kantorovich duality magically separates $\mathbb{P}_r$ and $\mathbb{P}_g$.

The optimal $f^*(x)$ can be approximated by the neural networks with parameters **W**, which is called the discriminator. Due to the approximation, the trained discriminator is not the optimal solution f^* even though it eventually converges. Instead, it finds the optimal solution in the subspace depending on network's architecture. Thus, the architecture affecting the solution deserves more investigations. Furthermore, when we represent discriminator as neural networks, K-Lipschitz condition should be guaranteed. The naive way is the weight clipping: restricting all weights within $[-c, c]$. Weight clipping is a terrible choice that can be improved by adding the gradient penalty, which becomes WGAN-GP discussed later.

Generator in WGAN The generator, denoted by $x = g_\theta(z)$, can be inserted into $f^*(x)$, which is assumed to be the optimum found by the discriminator in the previous step.

$$\arg\min_\theta \ W(\mathbb{P}_r, \mathbb{P}_g) = \arg\min_\theta \mathbb{E}_{x \sim \mathbb{P}_r}[f^*(x)] - \mathbb{E}_{z \sim \mathbb{P}_z}[f^*(g_\theta(z))] \tag{10}$$

Since the job of generator is the rival of discriminator, that is, minimizing EMD with respect to θ, the first term of Eqn.10 becomes a constant.

$$\arg\min_\theta W(\mathbb{P}_r, \mathbb{P}_g) \quad = \quad -\arg\min_\theta \mathbb{E}_{z \sim \mathbb{P}_z}[f^*(g_\theta(z))] \tag{11}$$

Therefore, we use another network representing the generator to find the optimal θ^*. The discriminator and the generator is alternatively trained, as 'adversarial' means. In practice, people prefer a unbalanced training for these two networks. Berthelot even came up with Boundary Equilibrium Generative Adversarial Networks, using a dynamic ratio during the training [17]. In our cases of WGAN and WGAN-GP, We use 4 times discriminator as most people did.

Comparison between GAN and WGAN Fig. 2 is a proof of concept that the flat response of discriminator causes vanishing gradient, making generator hard to train in real training scenario. It is a reproduction of Fig. 2 in [10]. The blue and orange histograms represent the distributions of real and fake training data points. The green curve is the discriminator score of each x. During generator training, if a training point is at, for example, $x = 4$, both discriminators give low scores, meaning that the data point is fake. However, GAN discriminator has 0 gradient at $x = 4$, so the generator does not know how to improve. On the other hand, WGAN discriminator provides useful gradient information at $x = 4$, so the generator knows it should move towards negative x direction to generate more realistic samples.

3.3 Wasserstein Generative Adversarial Networks with Gradient Penalty (WGAN-GP)

WGAN enforces the 1-Lipschitz constraint on the discriminator by weight clipping using empirically selected interval $[-c, c]$. The resulting set of discriminator function is a subset of K-Lipschitz

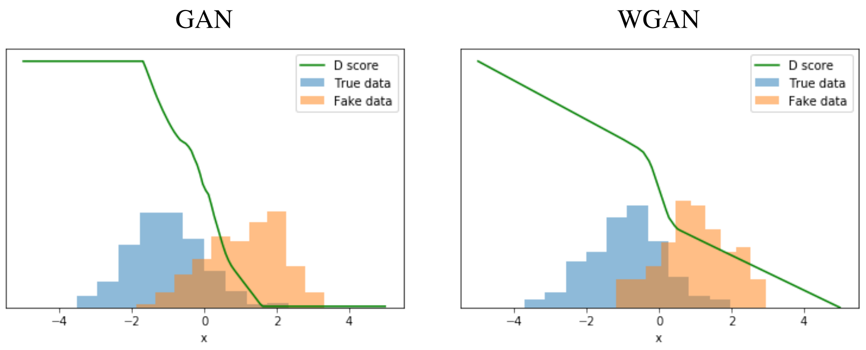


Figure 2: Left: an illustration of vanishing gradient in GAN discriminator. Right: in WGAN, the discriminator provides useful information, so the generator can be trained properly. The detailed description of how we generated the figure is in Appendix.

functions, where K depends on c and the discriminator's structure. The weight clipping approach of constraint implementation ends up learning very simple functions which neglect the higher moments of the distribution. The training process of WGAN also suffers from exploding or vanishing gradient when weight clipping range is not carefully tuned because of the interaction between the cost function and clipping constraint.

From **Corollary 1** in [11], the optimal function $f^*(x)$ optimizing $\max_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)]$ has gradient norm 1 at almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_g$. This insight leads to the implementation of a soft gradient penalty norm to constrain the gradient norm to be close to 1. The new loss is as follows,

$$L = \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)] + \lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}}D(\hat{x})||_2 - 1)^2] \tag{12}$$

where $\hat{x}$ is uniformly sampled from the straight lines connecting samples generated by $\mathbb{P}_r$ and $\mathbb{P}_g$, following **Proposition 1** in [11] that the optimal function $f^*(x)$ has gradient norm of 1 at points between points from $\mathbb{P}_r$ and $\mathbb{P}_g$.

We trained both WGAN and WGAN-GP on a toy experiment to demonstrate the advantages of WGAN-GP compared with WGAN with weight clipping to $[-0.1, 0.1]$. Fig. 3 shows the value surfaces of the discriminator and the weight distributions of WGAN and WGAN-GP. For WGAN-GP, the discriminator was able to accurately discriminate real and fake data as the value surface aligned well with the real data distribution, while the discriminator of WGAN was simple and failed to discriminate the two distributions well. The reason lies in the fact that weight clipping pushes the weights towards the clipping boundaries during training which reduces the expressiveness of the discriminator (Fig. 3 third column).

4 Experimental Setup

4.1 Dataset and Pre-processing

We use **Carvana Image Masking Challenge** (`https://www.kaggle.com/c/carvana-image-masking-challenge`) as our dataset. This dataset provides high-resolution vehicle figures from different angles for different cars. It also provides masks to filter out the background in the train set. If we combine the original figure and mask together, the main body of a vehicle can be captured.

With equipment limitation, only low resolution figures are used in our GAN training. In this project, image size of 64×64 is selected as the target resolution of training figures. The following steps are applied to the training data in the dataset as pre-processing.

1. Apply masks to original figures, and extract the vehicles.

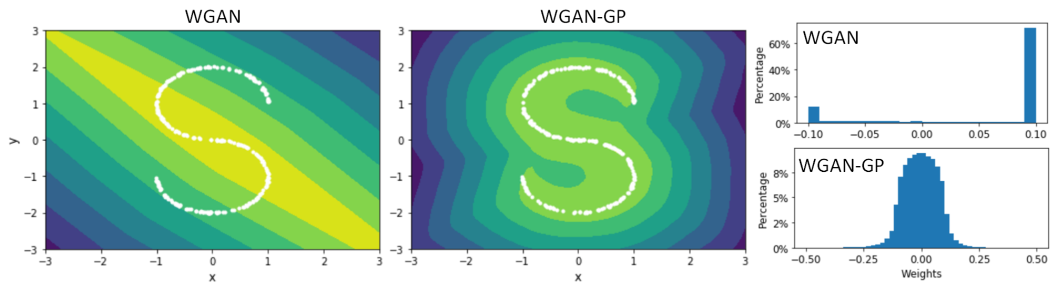


Figure 3: Value surfaces of the critics of WGAN (first column) and WGAN-GP (second column) in a toy experiment with an S-shaped real data distribution. The real data of the last iteration was depicted by the white points in both cases. The third column shows the weight distributions of the discriminators of WGAN and WGAN-GP. The detailed description of how we generated the figure is in Appendix.

2. Crop blank space to vehicle body. Finally select a 640×640 square mask because it can cover all the car images.
3. Resize the images to 64×64 pixels using a Python package called `skimage`.

Fig. 4 demonstrates the basic workflow for the pre-processing and some examples of the processed images. With the steps above, all input figures are transformed into 64×64 figures only containing the vehicle body and white background. As a result, 5088 pre-processed figures were obtained, and this set was used as the GAN training dataset.

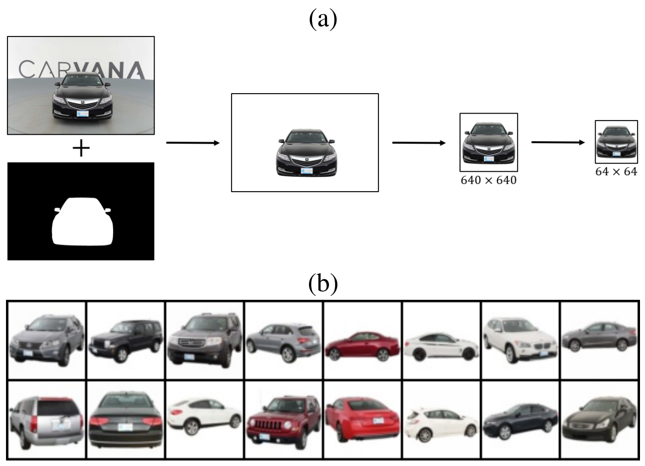


Figure 4: (a) Workflow of pre-processing the dataset. (b) Examples of the processed images used as real images in GAN training.

4.2 GAN Structure

To test the stability of the GANs, we trained 3 different network structures for different GAN losses. Specifically, we trained DCGAN generator and discriminator with and without batch normalization, and MLP generator and discriminator.

- **DCGAN generator**: For DCGAN generator without batch normalization, it consists of 5 layers of transpose convolution with kernel size 4×4, which takes in a noise vector of dimension 128 and upsamples it to an image of size $64 \times 64 \times 3$ (Fig. 5). The activation map size is doubled at each layer, while the number of channels is halved (except for the last layer). Each layer except for the last one is followed by a ReLU nonlinearity, and the

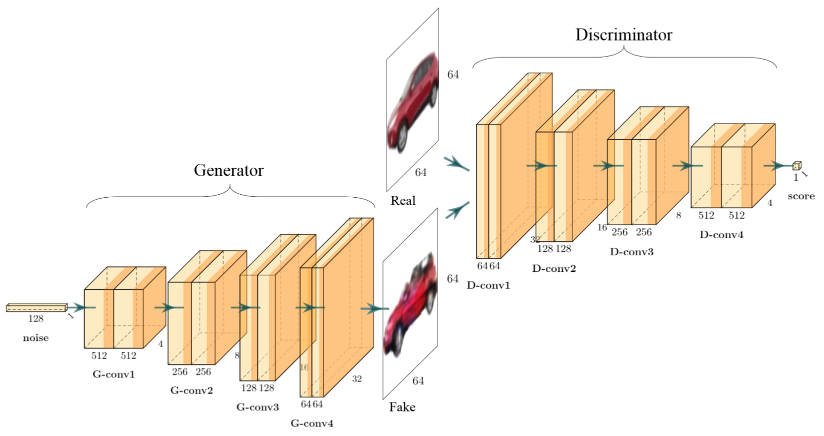


Figure 5: The network structure with DCGAN as the generator and discriminator.

last layer is passed through a tanh nonlinearity. Batch normalization is added after the first 4 layers in the case of DCGAN generator with batch normalization.

- **MLP generator**: For MLP generator, it consists of 5 fully connected layers with the first four passing through ReLU nonlinearity and the last one going through tanh nonlinearity.The number of units for the first layer to the last one is 512, 512, 1024, 2048, and 12288 respectively. The final output is reshaped to $64 \times 64 \times 3$ as the fake samples.

- **DCGAN discriminator**: For DCGAN discriminator for WGAN-GP and WGAN, it consists of 5 layers of convolution layers with the first 4 followed by LeakyReLU with slope parameter 0.2. The kernel size of each layer is 4×4. The stride is 2 for the first 4 layers, which, as a result, halves the input size by zero padding each dimension by 1. The last layer generates the discriminator score (Fig. 5). For GAN training, the score is further confined within (0, 1) with a sigmoid function.

- **MLP discriminator**: For WGAN-GP and WGAN training, MLP discriminator is comprised of 5 fully connected layers with 4096, 2048, 1024, 512, and 1 units. As in the DCGAN discriminator, the first 4 layers are followed by LeakyReLU with parameter 0.2.

4.3 Code Availability

WGAN-GP paper [11] provides their code online (`https://github.com/igul222/improved_wgan_training`), which is written in Python and TensorFlow. There are other open source GAN implementations, such as a PyTorch implementation (`https://github.com/caogang/wgan-gp`). During the implementation, we referred to their generator and discriminator structures, and built up the whole working pipeline of GAN series by ourselves from scratch.

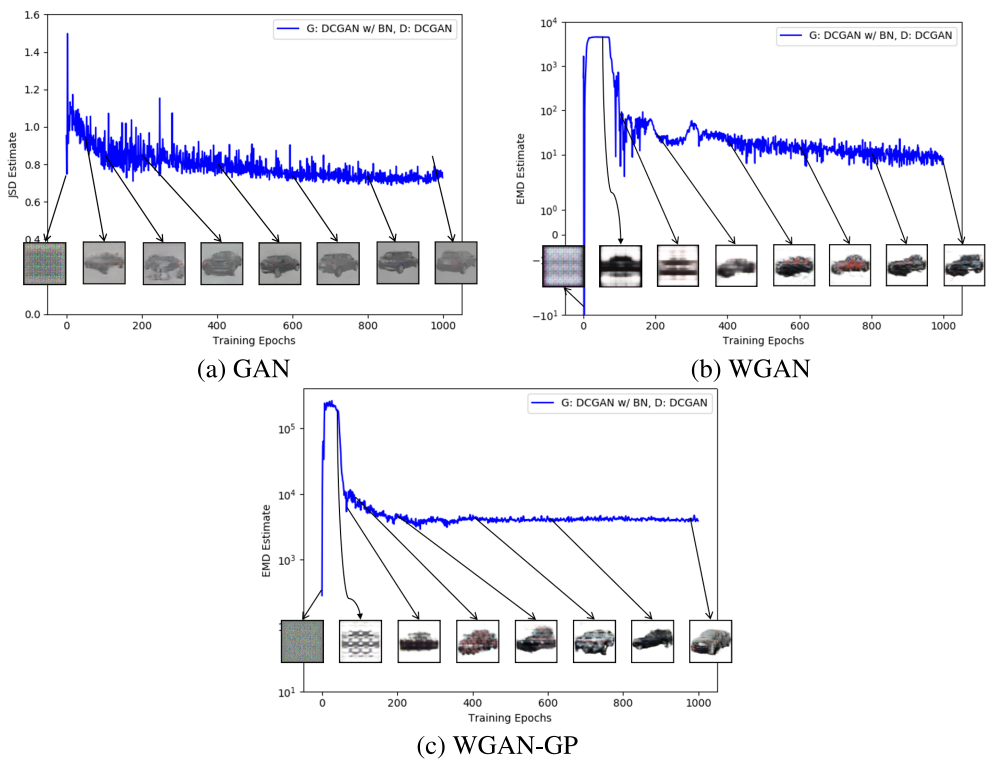
4.4 Training

Table 1 summarizes the parameters we used for different GAN losses. One challenge we faced during implementation is that GAN training is so slow. It typically takes hours to train one model. We did not tune the training parameters too much due to the limited project time, while we still obtained decent results. We did not change the parameters when we used other network structures for a specific GAN loss.

All experiments were caried on a machine equipped with graphic card NVIDIA GeForce GTX 1070, and CPU Intel(R) Core(TM) i7-8750H @ 2.20GHz. All experimental programs were implemented in PyTorch 1.1.0 and PyTorch-Vision 0.4.1, with CUDA 9.0 and CUDNN 7 enabled. The code shows in Appendix.

Table 1: Training parameters for GAN models.

	GAN	WGAN	WGAN-GP
Batch size	256	256	256
Weight clipping threshold	/	$[-0.01, 0.01]$	/
Penalty tuning parameter λ	/	/	10
Number of epochs	1000	1000	1000
Number of D training steps per G step	1	4	4
Optimizer	Adam	RMSprop	Adam
Learning rate	0.0002	0.0002	0.0002



(a) GAN (b) WGAN (c) WGAN-GP

Figure 6: JSD or EMD estimates of GAN, WGAN and WGAN-GP with G:DCGAN w/ BN, D:DCGAN.

4.5 Evaluation

The performance of GANs is evalueated in three aspects. 1) Inspection: We inspect and compare the quality of generated images to subjectively judge the performance of GANs. 2) Loss: we observe and compare the tendency of loss to compare the performance of GANs. 3) Computational cost: Given the same epoch and samples, we compare the cost of time to compare the performance of GANs.

5 Results

5.1 Loss and Performance

Fig. 6 presents the losses of GAN, WGAN, and WGAN-GP when the generator is DCGAN with batch normalization and the discriminator is DCGAN. The result is very consistent with our expectation in general: all the curves jump up to a high value at the beginning but decrease to a stabilized value. Here we account for the special features in Fig. 6.

- **Negative value of EMD in (c)**: in theory, EMD must be positive all the time, whereas we use neural networks to approximate $f^*(x)$. Thus, the loss, the proximate result, is possible to be negative.
- **WGAN-GP has a large loss (EMD) than WGAN**: note we exclude the gradient penalty for the curve fo WGAN-GP, it can be compared dirctly to WGAN. The large value of EMD in WGAN-GP is well explained by the bias led by the additional gradient penalty.
- **The loss dramatically rises up at the beginning**: since the discriminator learns to maximize the loss but generator learns to minimize it in reverse. The increments of loss means the discriminator outperforms the generator. It is because of 1) the structure of networks and 2) the unfair proportion of the iteration. After discrimanor is satuarated, the loss begins to decrease due to the domination of the generator.
- **The loss is asymptotically stabilized to a constant**: the loss never reaches zero because 1) the neural networks is a proximate solution that results in the bias, and 2) regularization operation including clipping and penalty introduce more bias.
- **The quality of images associates with loss**: for WGAN and WGAN-GP, the image quality increases with loss descending. For GAN, however, such relationship does not strictly hold.

5.2 Generated Fake Images

Fig. 7 shows the fake images generated from different combinations of network structures and GAN losses. We have the following observations.

- **Negative effect of weight clipping in WGAN**: for fixed generator and discriminator architectures, WGAN-GP produces the best images in terms of visual quality and class diversity. WGAN produces ugly images. The reason is that the expressiveness of the discriminator with weight clipping is very poor, as illustrated in Fig. 3, so the generator training also becomes bad.
- **Mode collapse of GAN**: the GAN images look good with fine details, but GAN produces many repeated objects such as white or black cars from the same view angle. This is sometimes called "mode collapse" in machine learning literature [11]. Intuitively, the generator does cheating by simply projecting all the latent inputs to similar output images which have high scores. This is an undesired property in GAN training because we want the generator to sample the whole real image distribution, not merely some points.
- **GAN cannot learn the white background**: looking at Fig. 7(a)(d), we observe that GAN cannot even learn the white background. We think the reason is the vanishing gradient in GAN discriminators shown in Fig. 1 and Fig. 2. In other words, the discriminator can easily tell the difference between real and fake images, but it does not provide usable gradient information to guide the generator.
- **Stability of WGAN-GP**: if we compare Fig. 7(c) (f) and (i), we see that WGAN-GP results are very stable no matter what generator or discriminator structures we use. On the other hand, the performance of GAN depends heavily on the network structure.
- **The effect of batch normalization is not very obvious in our experiment**: the purpose of comparing DCGAN generator with and without batch normalization is that in practice we would expect batch normalization gives more stable training. However, in our experiment we did not observe too much degradation caused by the removal of batch normalization layers in the generator.
- **GAN produces higher resolution images than others**: although GAN suffers from mode collapse, it produces images with finer details and higher resolutions, especially in Fig. 7 (a) (d). This is a side effect of mode collapse, which leaves more room for the network to learn finer details. This might be an advantage of GAN over WGAN.

5.3 Computation Time

Table 2 shows the training time of different combinations of GAN losses and network structures. WGAN-GP runs slower than WGAN because it requires an extra step to calculate the weight gradients and back-propagate the penalty term. The training time of GAN is comparable to WGAN-GP.

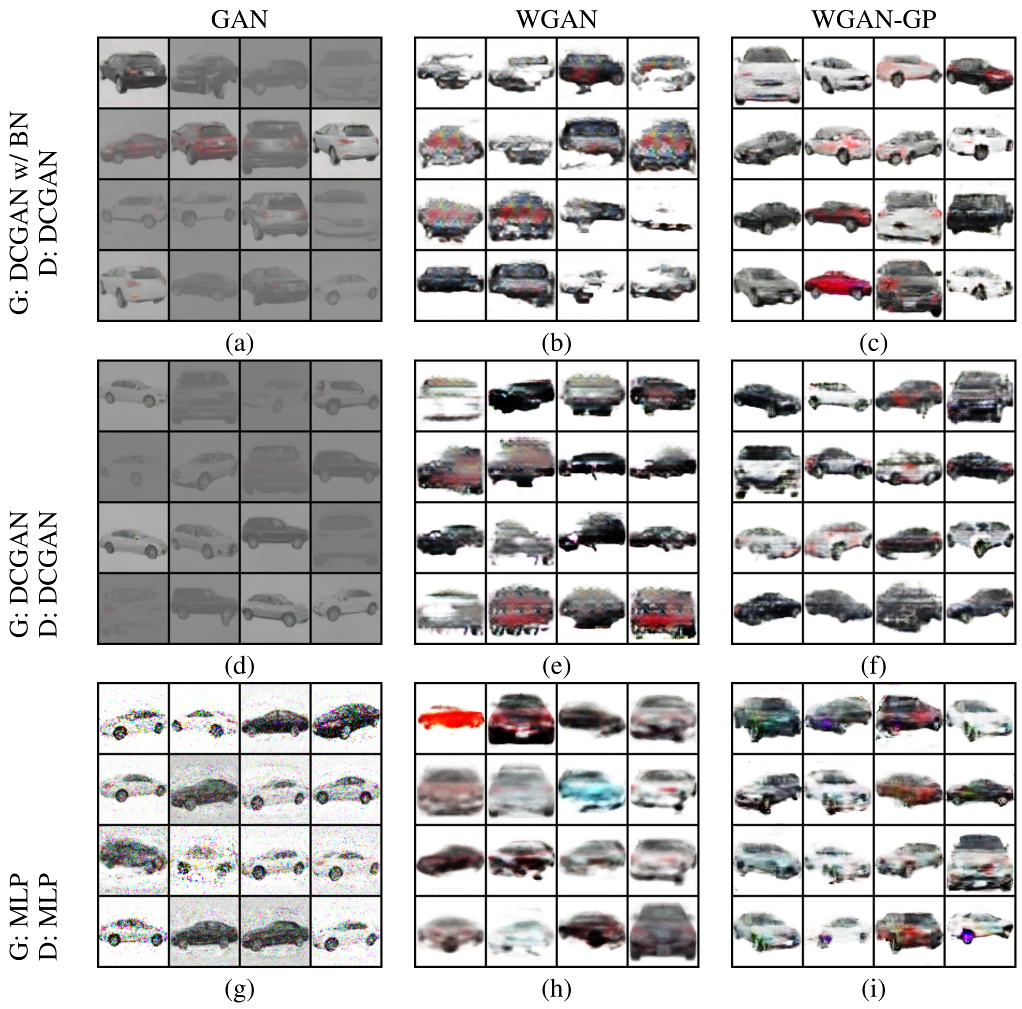


Figure 7: Fake images generated from different combinations of network structures and GAN losses.

Table 2: Training time of the GANs.

	GAN	WGAN	WGAN-GP
G: DCGAN w/ BN, D: DCGAN	6.5 hours	4.5 hours	5.5 hours
G: DCGAN, D: DCGAN	5 hours	4 hours	5 hours
G: MLP, D: MLP	5 hours	3.5 hours	5.5 hours

6 Conclusion

In this project, we have shown the different features of GANs and their derivation. The original GAN aims to minimize the JSD between the real data distribution and fake data distribution. WGAN introduces the discriminator gradient using EMD so as to improve the training stability of GAN. We illustrated this by simple examples. Moreover, the additional gradient penalty in WGAN-GP is a better option than weights clipping to enforce the K-Lipschitz continuity. We verify this statement by plotting the value surface of an S-shaped data.

As for the GAN training experiments, we demonstrated the feasibility of GANs to generating fake images. What the loss tells us is that the images produced are approaching the authentic images over epochs. According to the fake images, the best GAN loss is WGAN-GP, which is consistent with our expectation, and it is much more stable under various neural network architectures. In contrast, GAN relies on the architecture since it has mode collapse on both convolution models. Besides, the unexpected gray background is also a problem of GAN, and it remains to be solved as a future work.

We managed to accomplish our aims stated in the proposal for this project. However, we only scratched the surface of the GAN theories and did not dig into the rigorous mathematical foundations because of the limited time and our lack of mathematical knowledge. Instead, we did intuitive illustrations to demonstrate our understanding of the theories. Also, if time permits, finer parameter tuning during GAN training can be done.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. pages 1–9, 2014.

[2] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. *CoRR*, abs/1609.03552, 2016.

[3] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

[4] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[5] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised MAP inference for image super-resolution. *CoRR*, abs/1610.04490, 2016.

[6] Raymond A. Yeh, Chen Chen, Teck-Yian Lim, Mark Hasegawa-Johnson, and Minh N. Do. Semantic image inpainting with perceptual and contextual losses. *CoRR*, abs/1607.07539, 2016.

[7] Morteza Mardani, Enhao Gong, Joseph Y. Cheng, Shreyas Vasanawala, Greg Zaharchuk, Marcus T. Alley, Neil Thakur, Song Han, William J. Dally, John M. Pauly, and Lei Xing. Deep generative adversarial networks for compressed sensing automates MRI. *CoRR*, abs/1706.00051, 2017.

[8] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[9] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *CoRR*, abs/1611.02163, 2016.

[10] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. jan 2017.

[11] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. mar 2017.

[12] Martin Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. pages 1–17, 2017.

[13] Ming Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. *Advances in Neural Information Processing Systems*, pages 469–477, 2016.

[14] Ben Poole, Alexander A. Alemi, Jascha Sohl-Dickstein, and Anelia Angelova. Improved generator objectives for GANs. 2016.

[15] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[16] Jean-Paul Depretto. Des grèves en russie à la fin de l'union soviétique. *Le Mouvement social*, pages 3–8, 1994.

[17] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

A Experimental Setups of Simulations for Proof of Concepts

Simulating JSD and EMD of Two Gaussian Distribution The code is inspired by the paper of WGAN. There are two distribution that is actually discrete in a limited support. We design the experiments but refer to EMD distance MATLAB code provided francopestilli on github: https://github.com/francopestilli/life/tree/master/external/emd-2005-02. The KL divergence is provided by the 'kldiv' function from Mathworks .

Simulating the Discriminator Responses in 1D The training data points are sampled from 1D standard Gaussian distribution centered at $x = -1$ and $x = 1$, respectively, with batch size 1024. The discriminator architecture is 3 layers of multilayer perceptron (MLP) with ReLU nonlinearity and 512 units and without batch normalization. We trained the discriminators for GAN and WGAN for 10,000 iterations. During test phase, we take uniform samples from $[-5, 5]$ and get the response of the discriminators to form the score curve. This toy experiment is ran on Google Colab using Python 2 and TensorFlow 1.15.0. The code is based on the template code from https://github.com/igul222/improved_wgan_training.

Simulating the Value Surfaces of Critics The real data distribution $\mathbb{P}_r$ was an S-shaped curve with Gaussian noise of standard deviation 0.01. The generator distribution $\mathbb{P}_g$ was fixed in the experiment, which was the real data distribution with unit variance Gaussian noise added on it. Both discriminators were composed of 4 layers of multilayer perceptron (MLP) with ReLU nonlinearity and 512 units and without batch normalization. The gradient penalty coefficient was selected to be 0.1. Both networks were trained for 86,000 iterations on Google Colab using Python 2 and TensorFlow 1.15.0. The code is based on the template code from https://github.com/igul222/improved_wgan_training.

B Code

Our experimental code is published at Github https://anonymous.4open.science/r/a52ec4f5-0bd4-4264-9186-422f6dbe652b/