E0:270 - Machine Learning - Understanding Generative Adversarial Networks

Rajarshi Banerjee [1] **Arpan Mangal** [2]

Abstract

Using neural networks for generating synthetic data is a difficult task. Generative Adversarial Networks(GANs) provide an interesting approach to tackle this problem by pitting two networks to compete against each other. However training a GAN is an extremely difficult task. In this project we are trying to get an insight into the working of GANs and apply it to some interesting problem.

1. Problem statement

The aim of the project is to broadly survey a wide variety of GANs, to understand the various architectures, optimization techniques and/or other novel techniques used to better refine its training. We begin this exploratory journey from the humble beginnings of GAN (Ian J. Goodfellow et al., 2014) and follow its various advancements and pitfalls (Martin Arjovsky, 2017a) to quickly catapult ourselves to the bleeding edge of research on this topic (Martin Arjovsky, 2017b) only then to gently land on some of the aesthetic marvels produced by these networks (Jun-Yan Zhu et al., 2018) and finally suggest a small yet peculiar mutation to one of its variants.

2. Introduction

The advent of GAN came about with Ian Goodfellow's famous paper (Ian J. Goodfellow et al., 2014), and has enjoyed immense popularity ever since, at the heart of the matter lies a very simple formulation:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

Where $p_z(z)$ is the prior of the input noise variable, $p_{data}(x)$ is the unknown data distribution from which the training samples (are assumed to) come from. G and D are the generator and discriminator networks respectively.
The paper proves that when the discriminator D is trained to optimality, training the generator G minimizes $JSD(P_{data}||p_g)$, i.e. the Jensen—Shannon divergence between the model's distribution and the data generating process.

3. Deep Convolution GANs

Out of the many variants in GAN architecture, the deep convolution GAN (DCGAN) is one of the most simple architecture that is widely used, (Alec Radford & Luke Metz, 2016) also provides a list of guidelines to follow that has been empirically know to give good quality outputs. The guidelines suggested by the paper are:

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

4. Problems training a GAN

1. **Hard to achieve Nash equilibrium:** (Tim Salimans et al., 2016) discussed the problem with GANs gradient-descent-based training procedure. Two models are trained simultaneously to find a Nash equilibrium to a two-player non-cooperative game, with each model updating its cost independently with no respect to another player in the game does not guarantee a convergence.
2. **Low-dimensional supports:** (Martin Arjovsky, 2017a) discussed the problem of the supports of p_{data} and p_g lying on low dimensional manifolds and how

[1]Department of Computer Science and Automation, Indian Insitute of Science, Bangalore [2]Department of Computer Science and Automation, Indian Insitute of Science, Bangalore. Correspondence to: Arpan Mangal <arpanmangal@iisc.ac.in>, Rajarshi Banerjee <rajarshib@iisc.ac.in>.

Midterm Project Report, E0:270, Machine Learning, 2018.

it contributes to the instability of GAN training.
The dimensions of many real-world datasets, as represented by p_{data}, only appear to be artificially high.
Because both p_g and p_{data} rest in low dimensional manifolds, they are almost certainly gonna be disjoint.
When they have disjoint supports, we are always capable of finding a perfect discriminator that separates real and fake samples 100% correctly.

3. **Vanishing gradients:** When the discriminator is perfect, we are guaranteed with $D(x) = 1, \forall x \in p_{data}$ and $D(x) = 0, \forall x \in p_g$. Therefore the loss function for the generator falls to zero and we end up with no gradient to update the loss during learning iterations

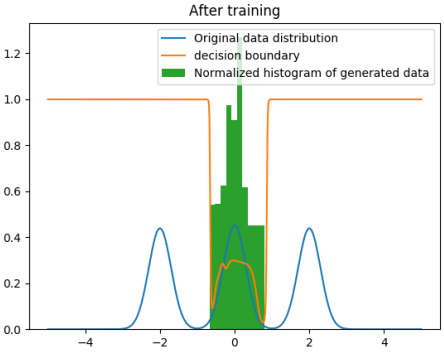


Figure 1: Effect on output distribution when discriminator is trained to optimality. Here the generator cannot learn the entire distribution as the discriminator is too powerful

4. **Mode Collapse:** During the training, the generator may collapse to a setting where it always produces same outputs. This is a common failure case for GANs, commonly referred to as **Mode Collapse**. Even though the generator might be able to trick the corresponding discriminator, it fails to learn to represent the complex real-world data distribution and gets stuck in a small space with extremely low variety.

5. **Lack of a proper evaluation metric:** Generative adversarial networks are not equipped with a good objective function that can inform us the training progress. Without a good evaluation metric, it is like working in the dark. No good sign to tell when to stop; No good indicator to compare the performance of multiple models.

5. Wasserstein GAN

As we mentioned, training of standard GAN is very unstable and slow. Main problem is with the loss metric i.e. JS

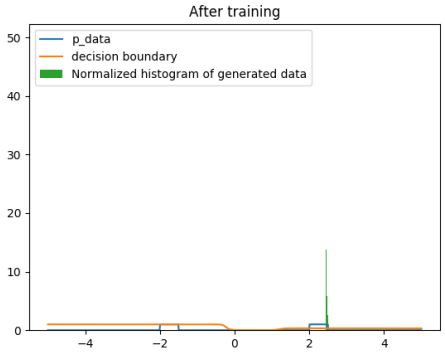


Figure 2: Effect on output distribution on training it on a mixture of two uniform distribution where generator is trained much more than the discriminator, here the output distribution collapses to a very small range. This scenario is known as mode collapse or helvetica

divergence, which is not continuous and differentiable everywhere. Due to which, it's very difficult to maintain balance between generator's and discriminator's training. If we train discriminator less, then our generator will not get true feedback to update, and on the other hand if we train it to optimality, we will hit vanishing gradient problem.
Recently there is a interest in the paper (Martin Arjovsky, 2017b), which claims to solve these problem. In WGAN, we use Earth-Mover(EM) distance as loss metric. EM distance under mild assumptions is continuous and differentiable everywhere. Assumption is that mapping of $\theta \rightarrow g(\theta)$ (where θ is generator's parameter) is continuous and **g** is locally Lipschitz, which holds in case of GAN's because g is multi-layer feed forward network.
It has following advantages over standard GAN's :

1. We can train our critic (same as discriminator) to optimality without vanishing gradient problem, therefore leads to correct feedback to generator and leads to stability.

2. EM distance is very meaningful loss metric i.e. correlates well with the quality of the generated samples.

3. The mode dropping phenomenon that is typical in GANs is also drastically reduced (empirically).

5.1. WGAN Formulation

The EM distance used here (also called as Wasserstein distance.

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} E_{(x,y)\sim\gamma}[\|x - y\|]$$

where $\gamma \in \Pi(p_r, p_g)$, is the set of all distributions whose marginals are p_r and p_g respectively.

In above formulation we have $\inf$ over all valid γ which is intractable. In paper, author cleverly convert it into $\sup$ using Kantorovich-Rubinstein duality. Intutively, it can be seen as dual of linear program (Herrmann, 2017).

Let $\mathbf{\Gamma} = \gamma(x, y)$ and $\mathbf{D} = \|x - y\|$, with $\mathbf{\Gamma}, \mathbf{D} \in \mathbb{R}^{l \times l}$

$$\text{EMD}(P_r, P_\theta) = \inf_{\gamma \in \Pi} \langle \mathbf{D}, \mathbf{\Gamma} \rangle_{\text{F}}$$

where, $\langle , \rangle_{\text{F}}$, is the Frobenius inner product (sum of all the element-wise products).
This can be thought as a minimization Linear program under the constraint the P_r and P_θ are marginals of γ.

Since, Infimium in this problem is intractable, we can solve its dual.

$$\text{EMD}(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_r} f(x) - \mathbb{E}_{x \sim P_\theta} f(x)$$

Suppose this function f comes from a family of K-Lipschitz continuous functions, $\{f_w\}_{w \in W}$, parameterized by w. In the modified WGAN, the discriminator model is used to learn w to find a good f_w and the loss function is configured as measuring the Wasserstein distance between p_r and p_g.

$$L(p_r, p_g) = W(p_r, p_g) = \max_{w \in W} \mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]$$

Thus the discriminator is not a direct critic of telling the fake samples apart from the real ones anymore. Instead, it is trained to learn a K-Lipschitz continuous function to help compute Wasserstein distance. As the loss function decreases in the training, the Wasserstein distance gets smaller and the generator models output grows closer to the real data distribution.

6. Cycle GANs

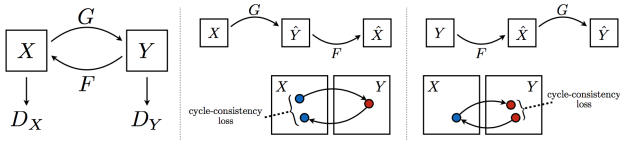


Figure 3: G and F are the generator networks that convert X to Y and Y to X respectively, with D_X and D_Y being the corresponding discriminators

Next we looked into some the practical applications of GANs in image transformations as discussed in the paper (Jun-Yan Zhu et al., 2018).

In a Cycle GAN we train two GANs in parallel to convert data from domain X to domain Y and vice-versa, and also add another loss called the cycle consistency loss that acts as the regularizer of the network.

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(x)) - y||_1]$$

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, X, Y) + \lambda \mathcal{L}_{cyc}(G, F)$$

6.1. Proposed modification to the Cycle GAN architecture

The architecture proposed in (Jun-Yan Zhu et al., 2018) is based on (Justin Johnson et al., 2016). We propose a simple modification to this architecture:
Rather than using two separate generators G and F we can use a common encoder and two decoders for each of the domains. The original architecture of the generator is a pair of convolution networks followed by 6 residual blocks and finally a pair of transposed convolution networks for upsampling. Our implementation uses a common encoder for both the GANs that consists of the first two convolution networks followed by 3 residual blocks, and two decoders for domain X and Y consisting of 3 residual blocks followed by 2 transposed convolution layers. It is essentially a partition of the original architecture into an encoder and a decoder.
This resulted in a much smaller size of the overall network and the training speed increased by more than a factor of 2, however the image quality was relatively poor.

7. Experiments

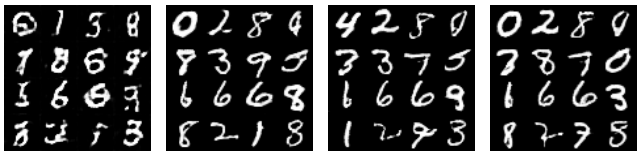
7.1. Dataset description

We used the MNIST dataset (Yann LeCun et al., 1998a) and LSUN outdoor church dataset (Yu et al., 2015) for training our DCGAN and WGAN. We used the datasets 'monet↔photo' , 'winter↔summer' as described in (Jun-Yan Zhu et al., 2018) for training the Cycle GAN and our modified variant of it.

7.2. Code

`https://github.com/Evil-Incorporated/basic-gans`

7.3. Results



(a) Epoch 1 (b) Epoch 10 (c) Epoch 20 (d) Epoch 25

Figure 4: DCGAN generator output on MNIST Dataset

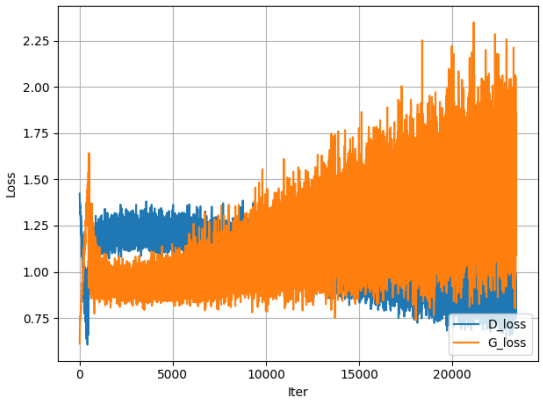


Figure 5: Discriminator loss D_{loss} and Generator loss G_{loss} on the MNIST dataset, this also goes to illustrate the fact that there is no proper evaluation metric in the standard GAN

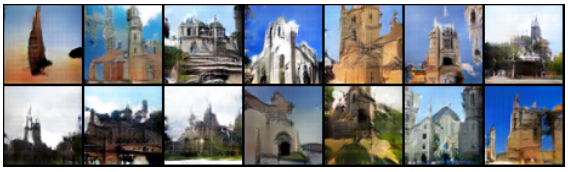
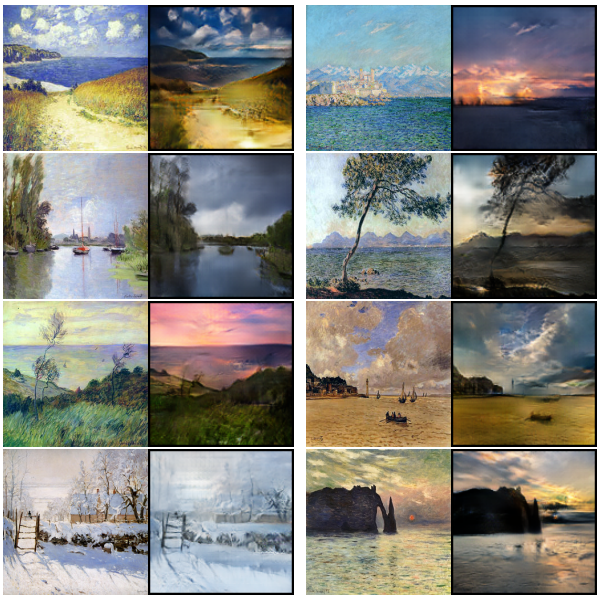


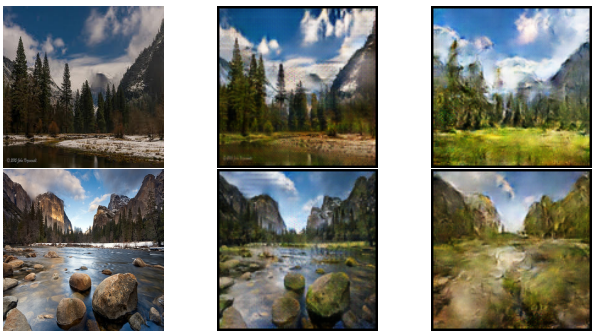
Figure 6: Output of DCGAN on LSUN outdoor church dataset after 25 epochs

Figure 7: Output of WGAN with DCGAN architecture on LSUN outdoor church dataset after 25 epochs



(a) Input (b) Ouput (c) Input (d) Ouput

Figure 8: Cycle GAN output of Monet paintings to photo-realistic pictures



(a) (b) (c)

Figure 9: Winter to summer conversion on the 'summer↔winter' dataset. (a) Input (b) Cycle GAN output (c) Fused encoder Cycle GAN output

8. Conclusion

In this paper we have presented a brief overview of the various aspects of GANs we have explored, by no means was this an exhaustive search and there are many more variants of GANs that we did not include in the project. More importantly we do not yet know whether our modification to the Cycle GAN is capable of generalizing to other domains, however we believe that it would be fruitful to direct our attention towards it in the future.

References

Alec Radford & Luke Metz, Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

Herrmann, Vincent. Wasserstein gan and the kantorovich-rubinstein duality, 2017.

Ian J. Goodfellow et al., Jean Pouget-Abadie, Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio. Generative adversarial nets, 2014.

Jun-Yan Zhu et al., Taesung Park, Phillip Isola Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2018.

Justin Johnson et al., Alexandre Alahi, Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

Martin Arjovsky, Léon Bottou. Towards principled methods for training generative adversarial networks, 2017a.

Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein gan, 2017b.

Tim Salimans et al., Ian Goodfellow, Wojciech Zaremba Vicki Cheung Alec Radford Xi Chen. Improved techniques for training gans, 2016.

Yann LeCun et al., Corinna Cortes, Christopher J.C. Burges. The mnist database, 1998a.

Yu, Fisher, Zhang, Yinda, Song, Shuran, Seff, Ari, and Xiao, Jianxiong. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.