Generative Adversarial Network 101[1]

Tong Li (tong.li@queensu.ca)
Julian M Ortiz (julian.ortiz@queensu.ca)

Abstract

Generative Adversarial Network, known as GAN, is one of the most promising architectures of generative models based on deep learning. The main purpose of the generative models is to generate synthesized data from the summarized distribution of given data. Therefore, finding the explicit or implicit approximation of the underlying real distribution of known data contributes the most to the ability of the generative models. With recent advances in deep learning algorithms, the impressive performance in recognizing the pattern of data further enhanced GAN's ability to synthesize data that fits in the distribution of real data. Such ability has great potential in the geoscience domain, in which the cost of new data is expensive. In this brief article, GAN, as a deep learning-based generative model within an adversarial framework, is introduced along with theoretical foundations, training procedures, and some variant structures of GAN.

1. Introduction

As we are going into the era of data, data science with deep learning algorithms are making striking achievements in lots of domains and changing our life. These data-driven methods have great abilities in discovering underlying hierarchical models that represent probability distributions over offered data (Bengio, 2009). The promise of such strong ability comes from a huge amount of input data training millions of parameters in these deep learning algorithms. In real applications like mining domains, data acquisition could be expensive and time-consuming. Researchers attempt to sidestep more data acquisitions by augmentations of known data, including translation, rotation, and flip (Krizhevsky et al., 2012). The diversity of data obtained by these minor modifications of data is relatively small. This motivates the utilization of generative models to produce synthetic data with more variability. Generative models approximate the real underlying probability distribution of input data and yield synthetic data within the best estimation of data distribution. However, difficulties in approximating many intractable probabilistic computations that caused by maximum likelihood estimation and related strategies hinder the performance of generative models (Goodfellow et al., 2014).

Generative Adversarial Network, as in its name, is a generative model trained in an adversarial net framework. GAN is normally composed of two networks, the discriminator and the generator. The discriminator is a classifier that tries to determine a sample more likely from real data distribution or GAN-modeled distribution and to separate synthetic data from real data. The generator learns to create synthetic data by incorporating feedback from the discriminator and tries to have a better estimation of the real data distribution to produce more realistic synthetic data. The basic principle of GAN is inspired by a two-player zero-sum game, in which each gain of utility is exactly the loss of the other player. The

[1] Cite as: Li T, Ortiz JM (2022) Generative Adversarial Network 101, Predictive Geometallurgy and Geostatistics Lab, Queen's University, Annual Report 2022, paper 2022-09, 132-140.

competition between the discriminator and the generator promotes them to get trained and improve their methods until no one could ever win the game. When the competition terminates at an equilibrium point of the discriminator and the generator, which is called Nash equilibrium in game theory (Ratliff et al., 2013), GAN can be considered to have captured its best approximation of the distribution of offered data and hence can generate new data within that modeled distribution. Theoretically, any differentiable function that maps data from one space to another can be used as the discriminator and the generator. GAN is initially implemented by multilayer perceptrons and commonly by variants of neural networks nowadays.

In the following sections, we will introduce some theoretical foundations of GAN, GAN implementations of these foundations, the structure and training procedures of GAN, and we will end with a brief review of variants of GAN.

2. Theoretical Foundation of GAN

2.1. Estimation on distribution

Generative models aim to discover the underlying distribution $P_{data}(x)$ of offered data (also called probability density function) which is composed by samples $\{x^1, x^2, \dots, x^n\}$. In real practice, $P_{data}(x)$ is usually unknown and extremely intricate. The goal of generative models is to find the best approximation of $P_{data}(x)$, which can be defined by a set of parameters θ, denoted as $P_G(x;\theta)$. By the way of finding the best approximation of $P_{data}(x)$, generative models can be classified into two classes: explicit density model and implicit density model. An explicit density model assumes the distribution and utilizes true data to train the model containing the distribution or fit the distribution parameters. An implicit density model produces synthetic data without an explicit distribution and uses produced data to modify the model.

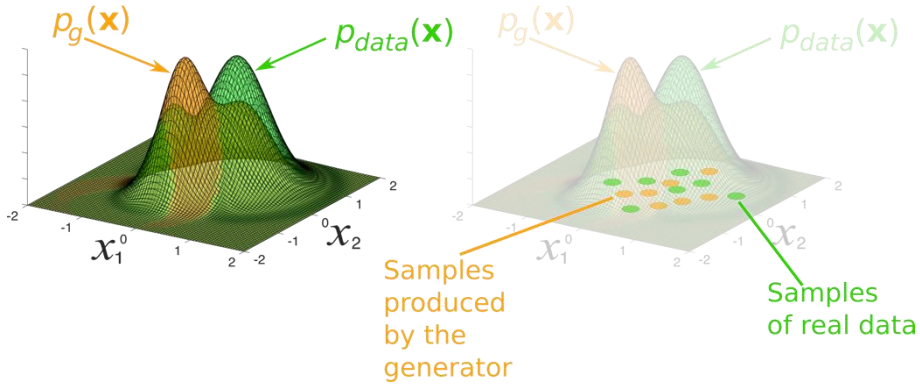

Figure 1: Illustration of estimation (orange) of real probabilities distribution (green) based on samples (Dumoulin et al., 2017).

To figure out how generative models find the best approximation of P_{data} utilizing a limited amount of real data, maximum likelihood estimation, as a representative of generative models, is illustrated. The likelihood function (Eq.1) is defined by the joint probability between real samples and the approximation distribution, which is used for the estimation.

$$L = \prod_{i=1}^{n} P_G(x^i;\theta) \tag{1}$$

To maximize the likelihood function, the best set of parameters θ^* is calculated as:

$$\theta^* = \arg\max_{\theta} L \tag{2}$$

The natural logarithm of the likelihood function is often used for calculation convenience. Since the logarithm is monotonic, the maximum of both functions occurs at the same value of θ.

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \ln \prod_{i=1}^{n} P_G(x^i;\theta) \\ &= \arg\max_{\theta} \sum_{i=1}^{n} ln\, P_G\left(x^i;\theta\right) \\ &\approx \arg\max_{\theta} E_{x\sim P_{data}}[\log P_G\,(x;\theta)]\end{aligned} \tag{3}$$

Eq. 3 is the main indicator for training both the generator and the discriminator in GAN, as well as where the adversary happens, which is going to be introduced in the next section.

If we go further on Eq. 3, minus it with the probability distribution of real samples in their own distribution, which is independent of θ, as

$$\begin{aligned}\theta^* &\approx \arg\max_{\theta} E_{x\sim P_{data}}[\log P_G\,(x;\theta)] \\ &= \arg\max_{\theta} E_{x\sim P_{data}}[\log P_G\,(x;\theta)] - E_{x\sim P_{data}}[\log P_{data}\,(x)] \\ &= \arg\max_{\theta} \int_x P_{data}\,(x)\log P_G(x;\theta)dx - \int_x P_{data}(x)\log P_{data}(x)dx \\ &= \arg\min_{\theta} KL(P_{data}(x)\|\,P_G(x;\theta))\end{aligned} \tag{4}$$

in which KL denotes Kullback–Leibler divergence. The KL divergence measures how two probability distributions are different from each other (Kullback and Leibler, 1951). By minimizing the KL divergence, the generative model will reach the set of parameters of the best approximation of underlying data distribution.

2.2. GAN-implementations of estimation

Estimation of the data distribution is done by fitting a known parametrized distribution to an underlying intractable distribution. One of the most powerful tools in fitting nonlinear functions, a neural network, is used in GAN to implement such estimation. The generator and the discriminator in GAN adopt two independent neural networks with adversarial purposes:

The generator neural network, denoted as ***G***, is defined by θ in Eq.3. ***G*** takes random input z from a predefined simple distribution like uniform and gaussian distribution $P_z(z)$ as input and represents a mapping to data space $G(z;\theta_G)$. The purpose of ***G*** is to yield synthesized data that cannot be detected by the discriminator.

The discriminator neural network, denoted as ***D***, is a discriminative model that outputs a single scalar $D(x;\theta_D)$ trying to separate synthesized data from real data. The best set of parameters and purpose of ***D*** can be defined as

$$\theta_D{}^* = \arg\max_{\theta_D}(\, E_{x\sim P_{data}}[\log D\,(x;\theta_D)] + E_{z\sim P_z}[\log(1 - D\,(G(z;\theta_G);\theta_D))]) \tag{5}$$

As a summary, G and D play the following minimax game with value function $V(G,D)$:

$$\min_G \max_D V(G,D) = E_{x\sim P_{data}}[\log D\,(x;\theta_D)] + E_{z\sim P_z}[\log(1 - D\,(G(z;\theta_G);\theta_D))] \tag{6}$$

Given any generator G, the value function $V(G,D)$ can be reformulated as below:

$$\begin{aligned} V(G,D) &= \int_x P_{data}\,(x)\log D\,(x;\theta_D)dx - \int_z P_z(z)\log(1 - D\,(G(z;\theta_G);\theta_D))dz \\ &= \int_x P_{data}\,(x)\log D\,(x;\theta_D) + P_g(x)\log(1 - D\,(x;\theta_D))dx \end{aligned} \tag{7}$$

For any $(a,b) \in \mathbb{R}^2\,\{0,0\}$, the function $a\log(y) + b\log(1-y)$ achieves its maximum in [0,1] at $\frac{a}{a+b}$. Thus, the optimal discriminator ***D*** is given by the maximum of $V(G,D)$.

$$D^*{}_G = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)} \tag{8}$$

With Eq. 8 and the KL divergence, the value function with a fixed ***G*** Eq.7 could be written as:

$$\begin{aligned} V(G,D) &= E_{x\sim P_{data}}[\log D^*{}_G\,(x)] + E_{z\sim P_g}[\log(1 - D^*{}_G\,(x))] \\ &= E_{x\sim P_{data}}\left[\log\frac{P_{data}(x)}{P_{data}(x) + P_g(x)}\right] + E_{z\sim P_g}\left[\log\frac{P_d(x)}{P_{data}(x) + P_g(x)}\right] \\ &= KL\big[(P_{data}\|(P_{data} + P_g)\big] + KL\big[\big(P_g\|(P_{data} + P_g)\big] \end{aligned} \tag{9}$$

Therefore, the value function of GAN is proven related to the KL divergence that measures the likelihood of two probabilistic distributions. Competition in this minimax game leads both the generator and discriminator to keep optimizing their methods. Finally, based on Sion's minimax theorem (Sion, 1958) and the proof in Goodfellow et al. (2014), two players ***G*** and ***D*** will converge at the same point, where $P_g = P_{data}$, then $G(z;\theta_G{}^*)$ can be considered as the best estimation of P_{data} and can be utilized to generate synthetic data.

3. Common structure and training of GAN

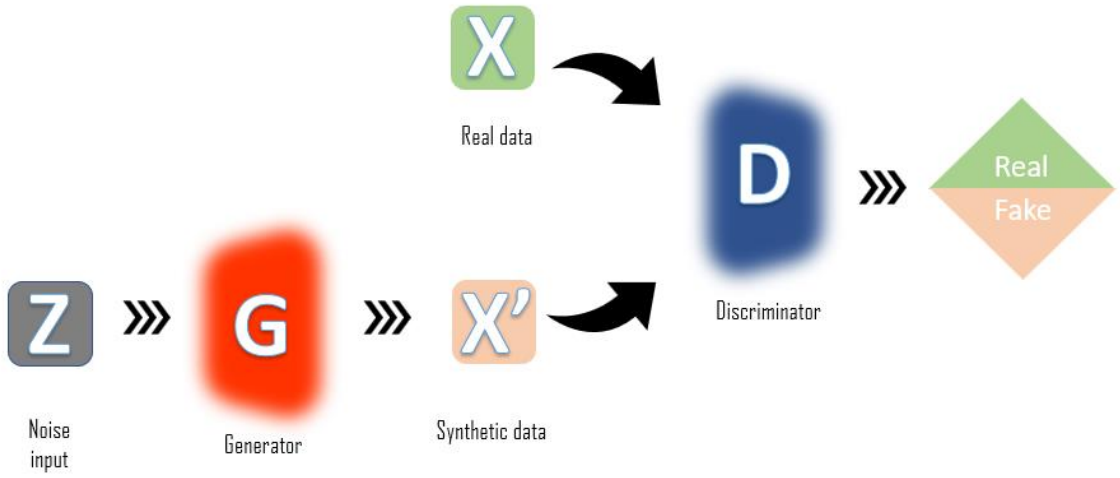

Figure 2: Typical GAN structure.

GAN is usually comprised of two core models, the generator and the discriminator. The generator takes random inputs and maps them into synthetic data with a differentiable function that can be trained by backpropagation. These random inputs are usually generated from a simple distribution like Gaussian distribution but can also be generated by certain rules in some variants of GAN for better estimation of the real data distribution or other purposes. The discriminator takes samples either from real data or from the generator and predicts a binary class label of real or synthetic. The results derived by the discriminator will be evaluated by ground truth and used for guiding optimization of the weaker model in the game.

Detailed training procedures are described as:

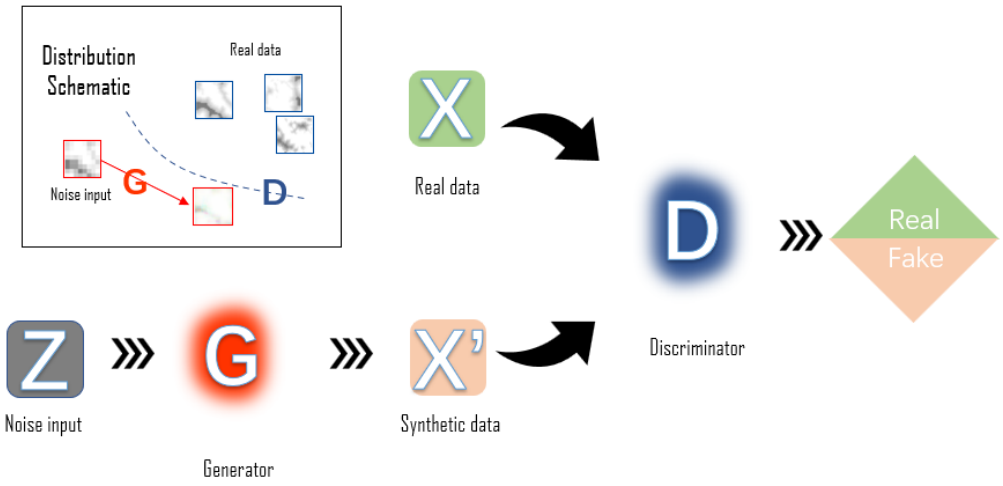

Figure 3a: The beginning of training a GAN.

Step one: At the very beginning, the generator has been set up randomly and hence the synthetic can easily be distinguished by the discriminator. The optimizer of GAN will train the generator by the gradients of the value function (or loss function in machine learning).

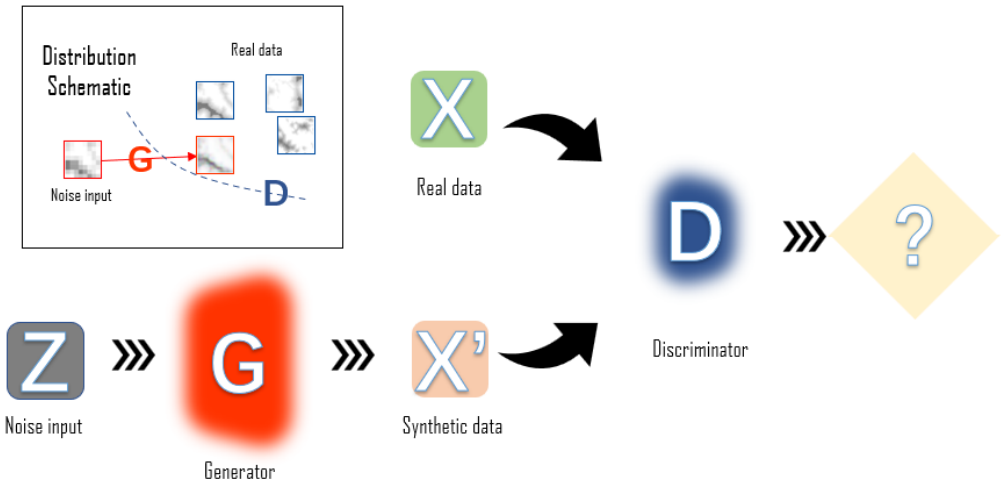

Figure 3b: After training the generator, the discriminator cannot decide which is fake.

Step two: Once the generator gets trained, the discriminator should be trained at least once. The reason for this procedure is a more optimized discriminator can provide a more accurate ratio between 2 distributions (see Eq. 8) and hence produce better gradients to train the generator.

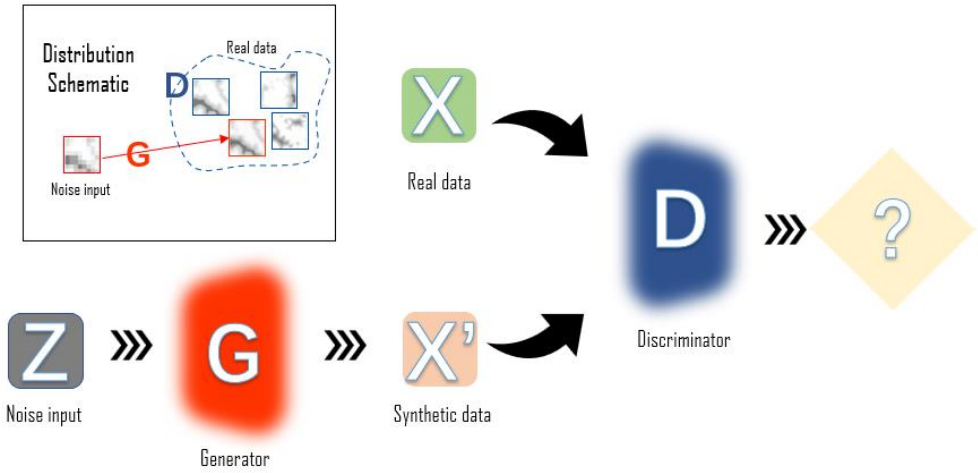

Figure 3c: In the final stage of the training of GAN, both the generator and the discriminator cannot be further trained.

Step three: Keep training with steps one and two, until the discriminator is maximumly confused and can not distinguish synthetic data from real data, which reflects in the prediction of 0.5 of any input sample either real or synthetic. Then the training is considered finished, and the synthetic data produced by the generator is lying in the distribution of real data.

The training procedure for GAN is usually challenging and unstable due to several reasons (see Radford et al.,2015 and Salimans et al., 2016). Certain useful tricks and improvements in the training of GAN are used:

1. Batch Normalization. Using a large batch of real or synthetic data after normalization for both the discriminator and generator will stabilize the training and speed up training.

2. High learning rate and more steps on training the discriminator. As mentioned in the training procedure, focusing more on the discriminator will provide with better gradient for the whole network.

3. Avoid hard boundaries. Deep neural nets are prone to producing highly confident outputs that identify the correct class but with too extreme of a probability (Goodfellow et al., 2016). Smoothing the boundaries of the activation functions by using leaky ReLU in the neural network (see Radford et al., 2015 for details) and using a one-side smoothed label (usually setting 0.9 instead of 1 on positive samples) would help the discriminator more efficiently on distinguishing of synthetic data (Salimans et al., 2016).

4. Variants of GAN

Since the original structure of GAN proposed by Goodfellow et al. in 2014, a huge number of GAN variants have been created. These variants of GAN are innovative for their improvement of structures, extensions of theory, or specific application-oriented design.

4.1. Wasserstein GAN

Instead of using KL divergence as the value function for optimization, Arjovsky et al. (2017) proposed Wasserstein GAN that uses the Earth-Mover distance for evaluating the distribution distance between real data and the synthetic data and uses a critic function supported by Lipschitz constraint to represent the discriminator. The advantage of using such Wasserstein distance is to stabilize the training of GAN by avoiding the gradient vanishing problem when the real data and synthetic data share very little overlap.

4.2. Deep convolutional GAN

The original structure of GAN uses multilayer perceptron (MLP) as the generator and discriminator. The limitation of MLP is all the data is treated the same and does not leverage the benefit of spatial structures as the convolutional neural network (CNN) does. Thus, CNN has way better performance at generating images than MLP. Deep convolutional GAN (DCGAN, Radford et al., 2016) substitutes the CNN for MLP in the original GAN and soon became the backbone of most of the variants of GAN. CNN structures in DCGAN utilize convolutional layers to capture features of data on different scales with convolutional kernels, the generator produces synthetic data with these features while the discriminator captures its own set of features that can be used to distinguish synthetic data from real ones.

4.3. Conditional GAN and Info GAN

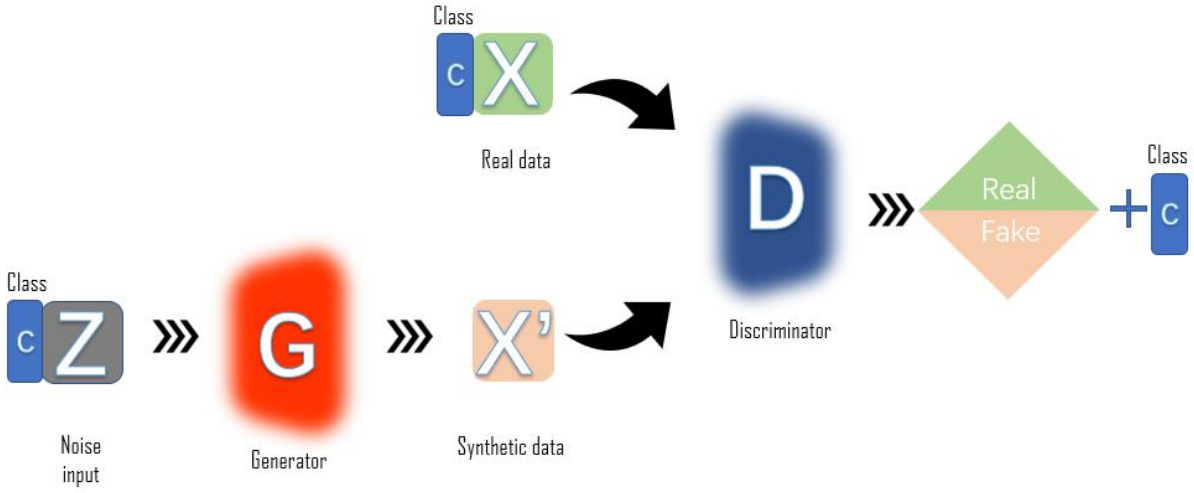

Figure 4: Intuitive structure of Info GAN.

The original GAN can only discover one data distribution and generate data in one estimated distribution. Conditional GAN (Mirza and Osindero, 2014) and Info GAN (Chen et al., 2016) add a "latent code" to the random noise input of GAN and adapt the generator and discriminator to this latent code in outputting class label of synthetic data. The value function of Info GAN is formulated as below:

$$\min_G \max_D \{f_I(D,G) = f(D,G) - \lambda I(c; G(z,c))\} \quad (10)$$

4.4. Super-resolution GAN

Super-resolution GAN (Ledig et al., 2017) takes low-resolution images as input instead of random noise and generates realistic details of the images while up-sampling. To achieve photorealistic super-resolution, SRGAN utilizes a perceptual loss function (notes as value function above) which consists of an adversarial loss to make sure the image manifold and a content loss to assure the perceptual similarity between real data and synthetic data other than simply pixel similarity. SRGAN uses ResNet (see He et al., 2016 for

details) in the generator, which is noted for its residual block that adds former output with its own output, called skip connection. ResNet in SRGAN is easier than training a CNN for capturing features and allows SRGAN to be substantially deeper to generate better results.

5. Conclusions

In this paper, we elaborate on the theoretical foundations, algorithm implementations, training procedures with useful tricks, and common variants of GAN. GAN is an implicit generative model that utilizes two neural networks in competition to generate synthetic data fitted in the underlying distribution of real data. The ability of GAN to generate infinite synthetic data from limited real data has great application values and potential in the geoscience domain.

6. Acknowledgments

We acknowledge the support of the National Natural Science Foundation of China (No. 42172326).

7. References

Arjovsky M, Chintala S, Bottou L (2017, July) Wasserstein generative adversarial networks. In *International conference on machine learning* (pp. 214-223). PMLR.

Bengio Y (2009) Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, *2*(1), 1-127.

Chen X, Duan Y, Houthooft R, Schulman J, Sutskever I, Abbeel P (2016) Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, *29*.

Goodfellow I (2016) Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Bengio Y (2020) Generative adversarial networks. *Communications of the ACM*, *63*(11), 139-144.

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90.

Kullback S, Leibler RA (1951) On information and sufficiency. *The annals of mathematical statistics*, *22*(1), 79-86.

Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Shi W (2017) Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4681-4690).

Mirza M, Osindero S (2014) Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Radford A, Metz L, Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

Ratliff LJ, Burden SA, Sastry SS (2013) Characterization and computation of local nash equilibria in continuous games. In Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on, pages 917–924. IEEE.

Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X (2016) Improved techniques for training gans. In Advances in Neural Information Processing Systems, pages 2226–2234.

Sion M (1958) On general minimax theorems. *Pacific Journal of mathematics*, *8*(1), 171-176.