Generative Adversarial Networks (GANs)

Amisi Fikirini, Elie Mulamba, Salomon Muhirwa, Lionel Nanguep, and Omer Elhussien

April 17, 2023

Abstract

During the previous decade, Generative Adversarial Networks have been evolving daily. It could be due to their ability to enrich models by providing a domain-specific approach. In the past, machine learning models needed more data available for training. For instance, several simple transformations were implemented in computer vision to improve the models. In this report, we briefly review GAN, discussing its architecture and addressing optimality for the generator and the discriminator. Furthermore, we present other types of GANs that focus on improving the loss function or conditioning the generator's and the discriminator's input. Finally, we implement Deep Convolutional GAN (DC-GAN).

Introduction

The limited amount of data could be a great challenge for machine learning models. It affects the model performance and increases its regularization error. For instance, approaches such as image flipping, cropping, and other simple transformations are used with image data in the field of computer vision to address this issue. With the arrival of GAN, people realized its tremendous advantage in providing a domain-specific approach for their models [1][2].

In this report, we review GAN, starting from its architecture, addressing the optimality problem, presenting several types of it, and finally discussing the Deep Convolutional GAN (DCGAN).

Generative Adversarial Networks

GAN is a deep-learning-based generative model. Its architecture involves two sub-models: the generator which generates new samples; the discriminator which classify the given input as from the data set (real), or from the generator (fake). Figure 1 presents this architecture.

The generator and the discriminator play a zero-sum game. In that game, if the discriminator successfully identifies real from fake data, it is rewarded, while the generator is penalized with extensive parameter updates. On the other hand, if the generator deceives the discriminator. The former will be rewarded, and the latter is penalized [3].

We have just witnessed adversarial learning. It is a process where two models try to weaken each other and, as a result, improve each other [4].

Model assumptions

Let G be a generator network which is a differential function represented by a multilayer perceptron (MLP) with parameter θ_g. Let z be a latent variable drawn from $\mathbb{P}(z)$. Then, $x = G(z)$ is a sample from the generator distribution $\mathbb{P}_G$.

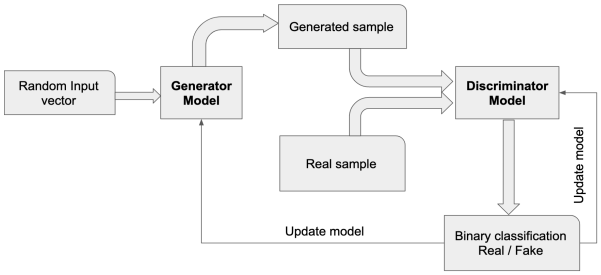

Figure 1: *GAN architecture.*

Let D be a discriminator network which a differential function represented by a MLP with parameter θ_d. Let x_i be a data point drawn from the data distribution $\mathbb{P}_{data}$ [5][6].

GAN objective function

During the training process, the generator G is trying to improve its ability to produce fake data points which tend to be similar to the real data set. On the other hand, the discriminator D tries to improve its classification ability with each round of training [7][8].

Simultaneously, train G and D with a minimax game:

$$\min_G \max_D \; V(G,D), \quad \textit{where} \tag{1}$$

$$V(G,D) = \left(\mathbb{E}_{x\sim\mathbb{P}_{data}}\left[\log D(x)\right] + \mathbb{E}_{z\sim\mathbb{P}(z)}\left[\log\left(1 - D(G(z))\right)\right]\right).$$

1. The discriminator wants $D(x) = 1$ for real data, and $D(x) = 0$ for fake data.
2. The generator wants $D(x) = 1$ for fake data.

Algorithm 1: GAN training algorithm

for *number of training iterations* **do**
  loop instructions
  (update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$,
  (update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$.

Optimality for the discriminator and the generator

Proposition 1 *For a fixed G, the optimal discriminator is,*

$$D^*_G(x) = \frac{\mathbb{P}_{data}(x)}{\mathbb{P}_{data}(x) + \mathbb{P}_G(x)}.$$

Theorem 1 *Let*
$C(G) = \mathbb{E}_{x\sim\mathbb{P}_{data}}\left[\log \frac{\mathbb{P}_{data}(x)}{\mathbb{P}_{data}(x)+\mathbb{P}_G(x)}\right] +$
$\mathbb{E}_{x\sim\mathbb{P}_G}\left[\log \frac{\mathbb{P}_G(x)}{\mathbb{P}_{data}(x)+\mathbb{P}_G(x)}\right]$.
The global minimum of $C(G) = -\log 4$ is achieved if and only if $\mathbb{P}_G = \mathbb{P}_{data}$.

The GAN research area is observing an explosion of publications each week. The types of GAN are beyond this report. We will try to present some of them such as:

1. Deep Convolutional GAN (DC-GAN).
2. Improved loss functions:
   - Wasserstein GAN (WGAN)
   - WGAN with gradient penalty.
3. Conditional GANS:
   - BigGAN.
   - Unpaired image-to-image translation: CycleGAN.

Deep Convolutional GAN architecture

DC-GAN is a direct extension of the GAN. It uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively [9].

The discriminator is made of strided convolutional layers, batch norm layers, and Leaky ReLU activations. The output is a single value probability that the given input is from the real data set. The input to this network is our original data set and the data from the generator. The strided covolution will provide better learning experience compared with using pooling, since it will offer a network to learn its own pooling function. Furthermore, batch norm and leaky relu are said to promote healthy gradient flow which is crucial part in the learning process [5].

The learning process is susceptible to usual machine learning problems such as gradient vanishing/exploding and under fitting. Furthermore, several techniques need to be implemented to ensure a stable training experience. For instance, the generator or the discriminator could over-fit if the other does not converge. Also, the model could collapse and the generator produces only limited sample varieties[1][10].

Related materials For GitHub implementation please check this link.

For mathematical proofs, please check this link.

References

[1] John Hany and Greg Walters. *Hands-On Generative Adversarial Networks with PyTorch 1. x: Implement next-generation neural networks to build powerful GAN models using Python*. Packt Publishing Ltd, 2019.

[2] Jason Brownlee. *Generative adversarial networks with python: deep learning generative models for image synthesis and image translation*. Machine Learning Mastery, 2019.

[3] Ian Goodfellow et al. "Generative adversarial nets ian". In: *Mining of Massive Datasets; Cambridge University Press: Cambridge, UK* (2014).

[4] Jakub Langr and Vladimir Bok. *GANs in action: deep learning with generative adversarial networks*. Manning Publications, 2019.

[5] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[6] Ian Goodfellow. "Nips 2016 tutorial: Generative adversarial networks". In: *arXiv preprint arXiv:1701.00160* (2016).

[7] Michigan Online. *Generative Models II*. `https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture20.pdf`. [Online; accessed 16-April-2023]. 2021.

[8] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[9] Kuan Wei. *Understand Transposed Convolutions*. `https://towardsdatascience.com/understand-transposed-convolutions-and-build-your-own-transposed-convolution-layer-from-scratch-4f5d97b2967`. [Online; accessed 16-April-2023]. 2020.

[10] Nathan Inkawhich. *DCGAN Tutorial*. `https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html`. [Online; accessed 16-April-2023]. 2023.