A deep introspection on Generative Adversarial Networks

Riccardo Lincetto, Guglielmo Camporese
Department of Information Engineering, University of Padova – Via Gradenigo, 6/b, 35131 Padova, Italy
emails: riccardo.lincetto, guglielmo.camporese @studenti.unipd.it

***Abstract*—GANs, namely Generative Adversarial Networks, are a hot topic nowadays.**

I. INTRODUCTION

Since its rise, deep learning had a great impact on discriminative models. Generative models instead were not affected by this innovation at first, but this trend changed with the introduction of Generative Adversarial Networks (GAN), a powerful framework first introduced in [1]. Since then, GAN gained more and more momentum because of the ability of training *deep generative models*, avoiding some of the difficulties encountered in other frameworks [2].

GAN is a sub-class of generative models where a probability density function (pdf) is implicitly defined; since GAN makes use of, generally two, *neural networks*, the pdf is induced by their architecture and parameters design: given a training set of sample data, distributed according to an unknown pdf p_d, the purpose of GAN is indeed to generate samples according to a distribution p_g, that mimics p_d, without explicitly defining it. As suggested by the name, this is achieved by putting in competition two entities: a generator and a discriminator. The task of the generator (G) is to generate data that can be regarded as true by the discriminator, while the discriminator (D) has the purpose of correctly distinguishing real from fake data. The classical real-life analogy with this process involves counterfeiters trying to produce fake currency and the police trying to detect it. This kind of interaction between the two entities can naturally be modeled with a game theoretical approach, where each player has its own strategies and payoffs, but in this paper we will rather talk about costs, as will be discussed in II. The major drawback of this framework anyway, is that the training of the model requires to compute the Nash Equilibrium (NE) of the game involving the two entities, which is not as simple as optimizing an objective function: without the guarantee of a NE, results obtained by the model could be different fom the ones desired.

In this paper we review some of the literature and explain our need to go back to the origins of GAN, implementing our own version of the code and simulating different scenarios, where in each one the discriminator is passed a different fake-to-true ratio of images.

The remainder of the paper is organized as follows: a brief overview of the literature is presented in II; a description of our work is then presented in III; the obtained results are presented in IV; finally we discuss our conlcusions in V.

II. RELATED WORK

Since their appearance in literature, GANs have been successfully applied to problems of image generation, editing and semi-supervised learning [5] [6]. *The results obtained with this technique were so good that they captured the attention of a great number of researchers, leading to a proliferation of various flavors of GAN, each performing better than the others on a specific domain.* It's difficult anyway to understand how to compare different GAN models, because of the lack of a consistent metric and the different architectures with which networks can be designed, which for each project are related to the corresponding computational budget. A tentative to define some guidelines to avoid these problems, together with a fair and comprehensive comparison of state-of-the-art GANs, is discussed in [3][MISSING ACCENTS ON REF]: what emerges here is that the computational budget plays a major role, allowing bad algorithms to outperform good ones if given enough time; plus, despite many claims of algorithms being superior to the original ones, there is no empirical evidence that they are across all datasets, in fact in most of them the original model outperforms those algorithms. [INSERT TABLE WITH LOSSES]. We thus report here a formalization of the original problem [1], modeling it with a game theory approach in a more precise way, as done in [4].

Deep Convolutional GAN (DCGAN) works as follows: the generator G maps an input noise random variable $z \sim U([-1, 1])$ to the data space as $G(z; \theta_g)$, where θ_g stands for the parameters of the network G. The discriminator network D defines a mapping between the data space and a scalar $D(x, \theta_d)$, where x can be an element either belonging to the training sample or generated by G, while θ_d represents the parameters of the network as before. $D(x)$ is defined as the probability of x being true data: the purpose of D is then to assign values close to one when $x \sim p_d(x)$ and close to zero when $x = G(z), z \sim p_z(z)$; G instead has the opposite task. We can infer this behaviors by setting appropriate loss functions:

$$L^{(D)} = -\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] - \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$
$$L^{(G)} = -L^{(D)}$$

where each player has to minimize its own loss. This competition can be naturally represented as a 2-player zero-sum infinite game: the player entities, G and D, are two neural networks, each with a given architecture defining its depth and width; pure strategies are the different combinations of weights for the given architecture, and it is immediate to notice that

these numbers are infinite; payoffs are defined as the opposite of their loss functions, which are also used for the training phase of the networks. An infinite game doesn't guarantee the existence of a NE, not even in mixed strategies, nor it allows to find it as the value of the game, i.e. maximinimizing the payoffs. As pointed out in [4] though, when using floating point numbers, the number of combinations of strategies becomes finite, albeit very large: there is then a NE, at least in mixed strategies, which however is hard to compute. It has to be kept into account also the fact that the optimization of a neural network can lead to a Local NE (LNE) because the problem is non-convex. In strategic form, GAN games can be described by a tuple $(\{G, D\}, \{S_G, S_D\}, \{u_G, u_D\})$, where S_G and S_D are the (finite) sets of strategies, whit elements θ_g and θ_d respectively, while $u_G = -L_G$ and $u_D = -L_D$ are the corresponding payoffs, for G and D respectively. The existence of a NE for this finite zero-sum game is granted can be found by maximinimizing its value:

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

To solve this problem, we can first fix a generator $\bar{G}$ and compute the optimal discriminator D, which is a simple optimization problem:

$$D^*(x) = arg\max_D \{V(D, \bar{G})\}$$

where, in continuous form, we can write

$$V(\bar{G}, D) =$$
$$\int_x p_{data(x)} \log(D(x))dx + \int_z p_z(z) \log(1 - D(\bar{G}(z)))dz =$$
$$= \int_x \Big[p_{data(x)} \log(D(x)) + p_g(x) \log(1 - D(x))\Big]dx =$$
$$= \int_x L(x, D(x), \dot{D}(x))dx.$$

If $D(x)$ is an optimal point it satisfies the Euler-Lagrange equation

$$\frac{\partial L}{\partial D} = \frac{dL}{dx}\frac{\partial L}{\partial \dot{D}}$$

and since $\partial L/\partial \dot{D} = 0$ we get:

$$\frac{\partial L}{\partial D} = \frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)} = 0$$
$$\Rightarrow \quad D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}.$$

The global minimum is found by setting $p_g = p_{data}$, in which case the value for the cost function becomes $-log(4)$.

It can be noticed that the payoffs are defined as the opposite of binary cross-entropy loss functions: this is particularly effective for discrimination tasks, but it's not for generation ones, because in the beginning of the training phase, when G is poor and D is able to correctly recognize generated samples, $log(1 - D(G(z)))$ *saturates* to zero thus resulting in a poor gradient. The learning in that case is too slow, but this problem can be avoided changing the loss function for G: instead of minimizing $log(1-D(G(z)))$, it can maximize $log(D(G(z)))$. This is formally defined as a Non-Saturating GAN (NSGAN), where the losses to be minimized are:

$$L^{(D)} = -\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] - \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$
$$L^{(G)} = -log(D(G(z)))$$

This new game isn't zero-sum any more, but the same equilibrium point of the dynamics can be found as before, thus providing the same theoretical results.

III. EXPERIMENTAL SETTING

For our project, we implemented a version of NSGAN which can be found at [**?**]. Usually the training of neural networks occurs after mini-batches of data are passed to them: for the discriminator network D, in a mini-batch there are the same number of true and of fake images. We instead carried out the training with different configurations, where D is passed different true-to-fake data ratios: this is done in practice defining a parameter α that represents the portion of true data with respect to the size of the mini-batch. This results in a parametrisation of the loss function as follows: [FOR D]

$$\alpha \cdot \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + (1 - \alpha) \cdot \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Performing the same analysis as before, we found that the optimal discriminator should have the form:

$$D^*_\alpha(x) = \frac{\alpha \cdot p_{data}(x)}{\alpha \cdot p_{data}(x) + (1 - \alpha) \cdot p_g(x)}$$

and that the loss of the generator is minimized, as before, when

$$p_g = p_{data},$$

from which the optimal D can be rewritten as

$$D^*_\alpha(x) = \alpha.$$

To support these results, it can be noticed also that when setting $\alpha = 0.5$, i.e. in the same case previously analysed with the same amount of true and fake images in a mini-batch, the results are consistent.

Before testing this model on MNIST dataset, we tried it on different ad-hoc 2D distributions, with the following parameters:

The results obtained with them are reported in IV, together with the same plot obtained for the MNIST dataset.

IV. RESULTS

V. CONCLUSIONS

REFERENCES

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[2] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[3] Mario Lu?i?, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. *arXiv*, 2017.

[4] F. A. Oliehoek, R. Savani, J. Gallego-Posada, E. van der Pol, E. D. de Jong, and R. Gross. GANGs: Generative Adversarial Network Games. *ArXiv e-prints*, December 2017.

[5] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[6] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *CoRR*, abs/1612.03242, 2016.