Appendix

Warith HARCHAOUI

Introduction

For dramatic computational speed improvements reasons, some pretraining techniques from 1990s are less popular nowadays in neural networks training. Recently, the rise of Generative Adversarial Networks (GANs) initiated by Goodfellow et al. [2014] is revisiting a new kind of optimization: not an usual minimization problem but a min-max problem hence the adjective *adversarial* which is thus more difficult to monitor. In practice, for same domain (or medium) data, it is possible to use the same structure and hyper-parameters as other authors e. g. the DCGAN structure by Radford et al. [2015] for images. Unfortunately, for custom data such as genomics data, there is a need for practical rule of thumbs to begin working for new kinds of data and neural networks and particularly wirh GANs.

Historically, neural networks experts used to train an unsupervised auto-encoder, in order to keep the encoder as an initialization of the neural network now trained in a supervised fashion. This pre-training technique seems to save a lot of time during an era of expensive and long computations. The aim of this appendix is to follow this kind of reasoning to initialize GANs that are notoriously painful to train and monitor without further data knowledge.

Similar Objectives

Let's study the Wasserstein GAN [Arjovsky et al., 2017] objective function with a generator $\mathcal{G}$, a real data distribution p living in $\mathbb{R}^D$, and noise generator distribution n living in $\mathbb{R}^d$

$$\min_{\mathcal{G}} W(p, q_{\mathcal{G}}) \tag{1}$$

where the Wasserstein distance $W(p, q_{\mathcal{G}})$ is defined between the distributions of real data p and generator data $q_{\mathcal{G}}$ (defined by $\mathbf{y} \sim q_{\mathcal{G}} \iff \mathbf{z} \sim n$, and $\mathbf{y} = \mathcal{G}(\mathbf{z})$)

$$W(p, q_{\mathcal{G}}) = \min_{\gamma \in \Gamma(p, q_{\mathcal{G}})} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma} \left[\|\mathbf{y} - \mathbf{x}\|_2 \right] = \min_{\gamma' \in \Gamma(p, n)} \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim\gamma'} \left[\|\mathcal{G}(\mathbf{z}) - \mathbf{x}\|_2 \right] \tag{2}$$

where $\Gamma(p, q)$ stands for the set of couplings γ based on the two marginal distributions p and q. In the GAN literature, this is reformulated by thanks to the Kantorovich-Rubinstein duality but in this appendix, we will still use the Wasserstein formulation.

Indeed, if we look at that auto-encoder objective function made of an encoder $\mathcal{E}$ and decoder $\mathcal{D}$:

$$\min_{\mathcal{E},\mathcal{D}} AE(p, \mathcal{E}, \mathcal{D}) \tag{3}$$

where the AE operator is simply the mean distance between the original data from p and the same data distorted by an encoder $\mathcal{E}$ followed by a decoder $\mathcal{D}$:

$$AE(p, p_{\mathcal{E},\mathcal{D}}) = \mathbb{E}_{\mathbf{x}\sim p} \left[\|\mathcal{D} \circ \mathcal{E}(\mathbf{x}) - \mathbf{x}\|_2 \right] \tag{4}$$

We believe that equations Eq. (2) and Eq. (4) are similar. Training an auto-encoder and training a generative adversarial network are both unsupervised but auto-encoder are easier to train because there is only a one-way minimization which is not the case for GANs that have two-ways type of minimization (min-max) w.r.t. different parameters.

By training an auto-encoder, we get a reasonable reconstruction loss for output functions $\mathcal{E}$ encoder and $\mathcal{D}$ decoder. Now, it is possible to estimate the mean $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{E}(\mathbf{x}_i)$ and covariance $\boldsymbol{\sigma} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{E}(\mathbf{x}_i) - \boldsymbol{\mu}\right)^2}$ (with element-wise square root and square functions). The intuition is that for a Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$, then the decoded points $\mathcal{D}(\boldsymbol{\mu} + \boldsymbol{\sigma} \times \boldsymbol{\epsilon})$ (with element-wise product) has a distribution that is close to decoded codes $\mathcal{D} \circ \mathcal{E}(\mathbf{x})$ which is close to the original data $\mathbf{x}$. This way, we made a reasonable initialization for a generator made of a Gaussian noise into a linear layer from empirical mean $\boldsymbol{\mu}$ and standard deviations $\boldsymbol{\sigma}$ followed by the decoder function $\mathcal{D}$.

Early on since the beginning of the GAN literature in 2014, deep learning scientists saw the potential of seeing the min-max optimization scheme as the minimization of a discrepancy between real and generated data where that discrepancy required and adversarial optimization for estimation [Goodfellow, 2016] (i. e. a maximization of an objective to be minimized *in fine*). Back to our work, at this point, the auto-encoder training was a traditional *one-way* minimization but now the adversarial network (or critic against the generator) is needed to measure the discrepancy we just mentioned between real and generated data. In our Wasserstein discrepancy case, this correponds to only maximization via the Kantorovich-Rubinstein duality of the Wasserstein distance. Once both the generator from the auto-encoder minimization and next the critic from the discrepancy maximization for estimation are done. We actually not only know how far our generator (i. e. decoder) combined with our noise generator is from the real data distribution but we also get a convincing initialization for regular GANs training.

To sum up, we propose three steps:

1. Auto-Encoder Minimization for the euclidean reconstruction error to be low;
2. Critic Maximization until real v. s. generated discrepancy has plateaued out;
3. Regular GAN Min-Max optimization with a generator initialized by a Gaussian fitting the encoded codes followed by the decoder on the one side and the critic previously maximizing the real v. s. generated discrepancy on the other side;

which is much easier for monitoring and prototyping as getting rid off an unpredictable objective variations behavior especially at the first epochs contributes to acceleration of development and robustness w.r.t. uncontrolled objective variations. In these conditions, finding hyper-parameters is much more simplified.

Experiments training DCGAN on CIFAR-10

Generative Adversarial Networks [Goodfellow et al., 2014] really attracted the research community attention when image rendering started to get convincing especially with the spark ignition DCGAN work of Radford et al. [2015] with a recent research and engineering accomplishment in extremely large scale settings of BigGAN [Brock et al., 2019] and mainstream press demos[1]. Following this trend, we use the same neural network structure as DCGAN [Radford et al., 2015] shown in Fig. 1 for a Wasserstein GAN trained on CIFAR-10 images dataset [**?**].

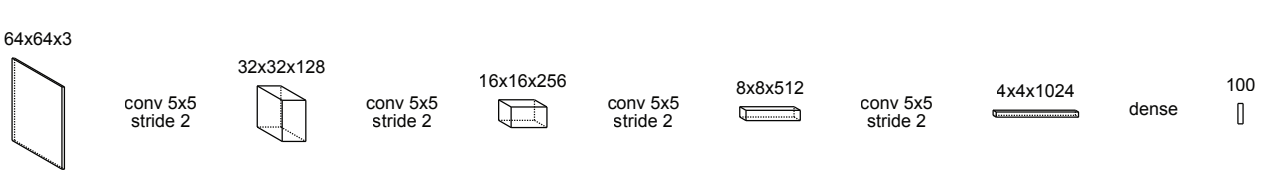

Figure 1: DCGAN encoder structure, the critic has same structure (with an additional 100-1 perceptron layer) and the decoder/generator has a symmetric structure

In this section, we use Adam [Kingma and Ba, 2014] default parameter with 10^{-6} for generator learning rate and 10^{-5} for critic learning rate and also 10^{-5} the one-way minimization learning rate.

[1] `https://www.thispersondoesnotexist.com`

Moreover, in min-max optimizations, we do 10 max iterations followed by 1 min iteration for each data mini-batch. The first step according to our section consists in training an auto-encoder. Fig. 2 is showing the evolution of the minimized loss w.r.t. the number of iterations.

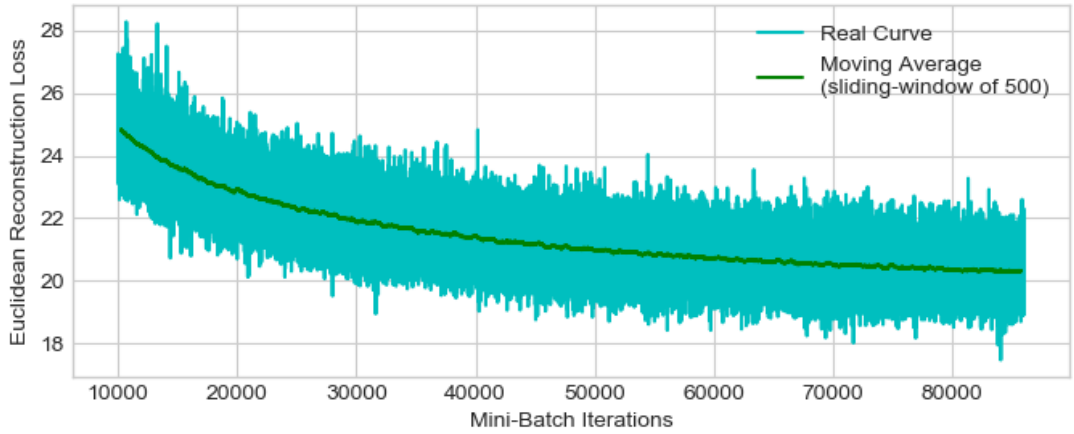

Figure 2: Autoencoder mini-batch loss w.r.t. the number of iterations (the lower the better, 50 iterations for 1 epoch)

Second step consists in estimating the mean μ and standard deviation σ of the codes coordinates (codes are encoded version of data). Then a critic maximization is taking place so that we can estimate the Wassertein distance between the real data and the decoded Gaussian noise associated with $\mathcal{N}(\mu, \sigma)$ as shown in Fig. 3. Here, the Gaussian parameter (μ, σ) and the decoder playing now the role of generator are all remaining frozen (i. e. with zero learning rate) so that the critic neural network can be considered as just a tool to estimate the Wasserstein distance between fixed paramaters noise and generator data distribution and real data distribution in a Kantorovich-Rubinstein maximization procedure.

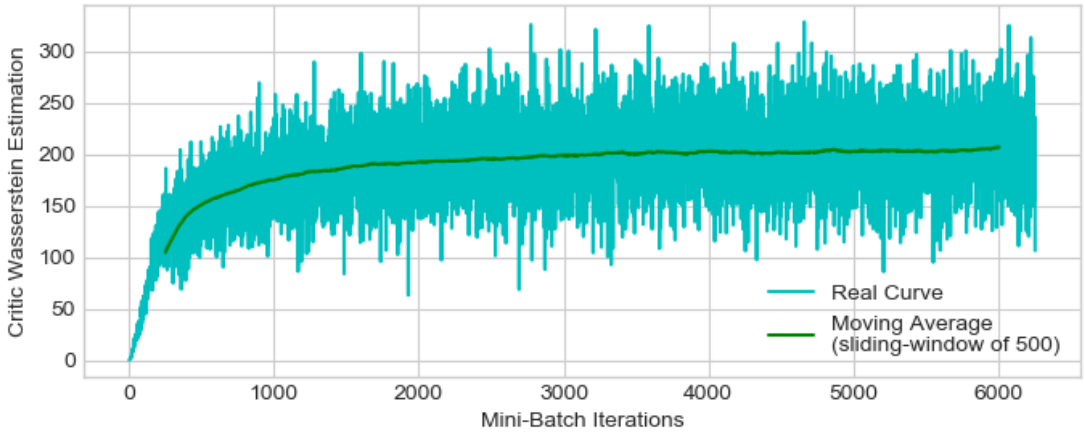

Figure 3: Critic Mini-Batch Wasserstein Distance Estimate w.r.t. the number of iterations. (the higher the better, 50 iterations for 1 epoch)

Third and final step is doing the original optimization scheme but now from a relevant starting point (which is crucial in non-convex deep learning optimization notoriously prone to spurious extrema). Fig. 4 is showing a direct minimization in spite of the adversarial min-max learning procedure. Here all parameters are allowed to vary meaning that the first two steps are just initialization steps following the *end-to-end* recommendations.

All these steps (except the first and easy auto-encoder step) are greatly simplified thanks to the ease of using the Lipschitz property for the critic neural network offered by Miyato et al. [2018] (this would not be reasonable with clipping [Arjovsky et al., 2017] nor regularization term [Gulrajani et al.,

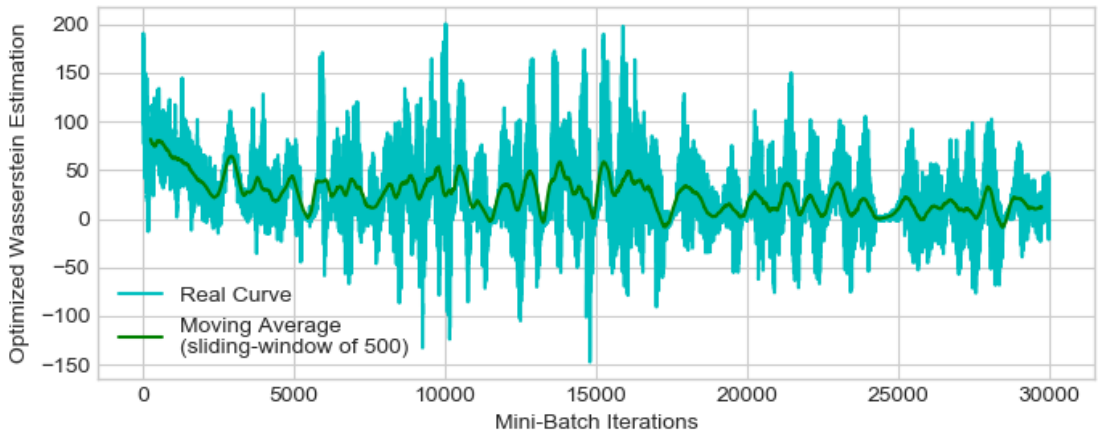

Figure 4: Our pre-trained GAN mini-batch objective w.r.t. the number of iterations. (the lower the better, 550 iterations for 1 epoch as each data mini-batch is seen 10 times for the critic and once for the generator)

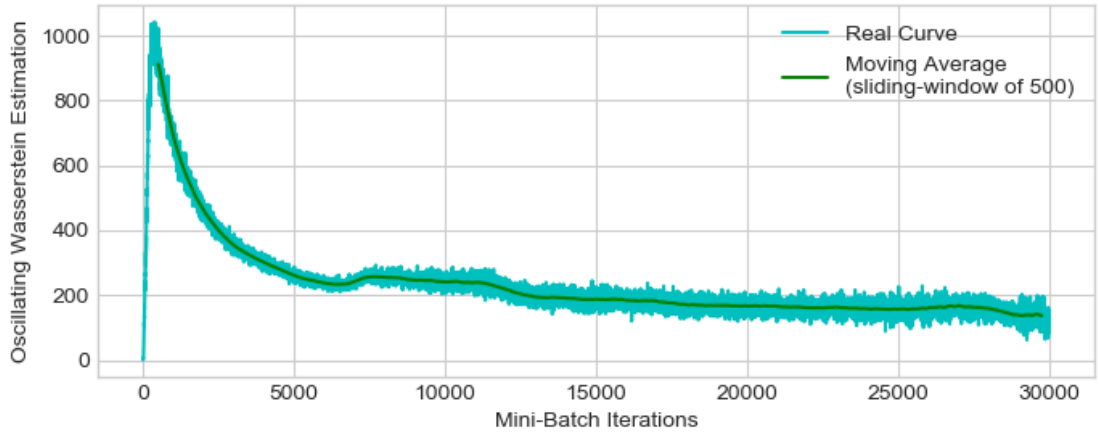

Figure 5: Oscillating GAN mini-batch objective w.r.t. the number of iterations. (the lower the better, 550 iterations for 1 epoch as each data mini-batch is seen 10 times for the critic and once for the generator)

2017]). Compared to direct GAN training shown in Fig. 6, with fewer iterations and less tedious selection of learning rates, our approach gets better results. Indeed, one-way optimization schemes are easier than two-ways min-max optimization ones to handle. With mastered tools described by LeCun et al. [1998] and decades of shared good practice, one-way optimization (say minimization and not min-max optimization) is well studied nowadays. This is to be put in contrast with the two opposite learning rates required to the direct stochastically alternated method. To be honest, in our case, the final step has the same min-max GAN-like algorithm but the variations are better controlled in practice because the system is better initialized.

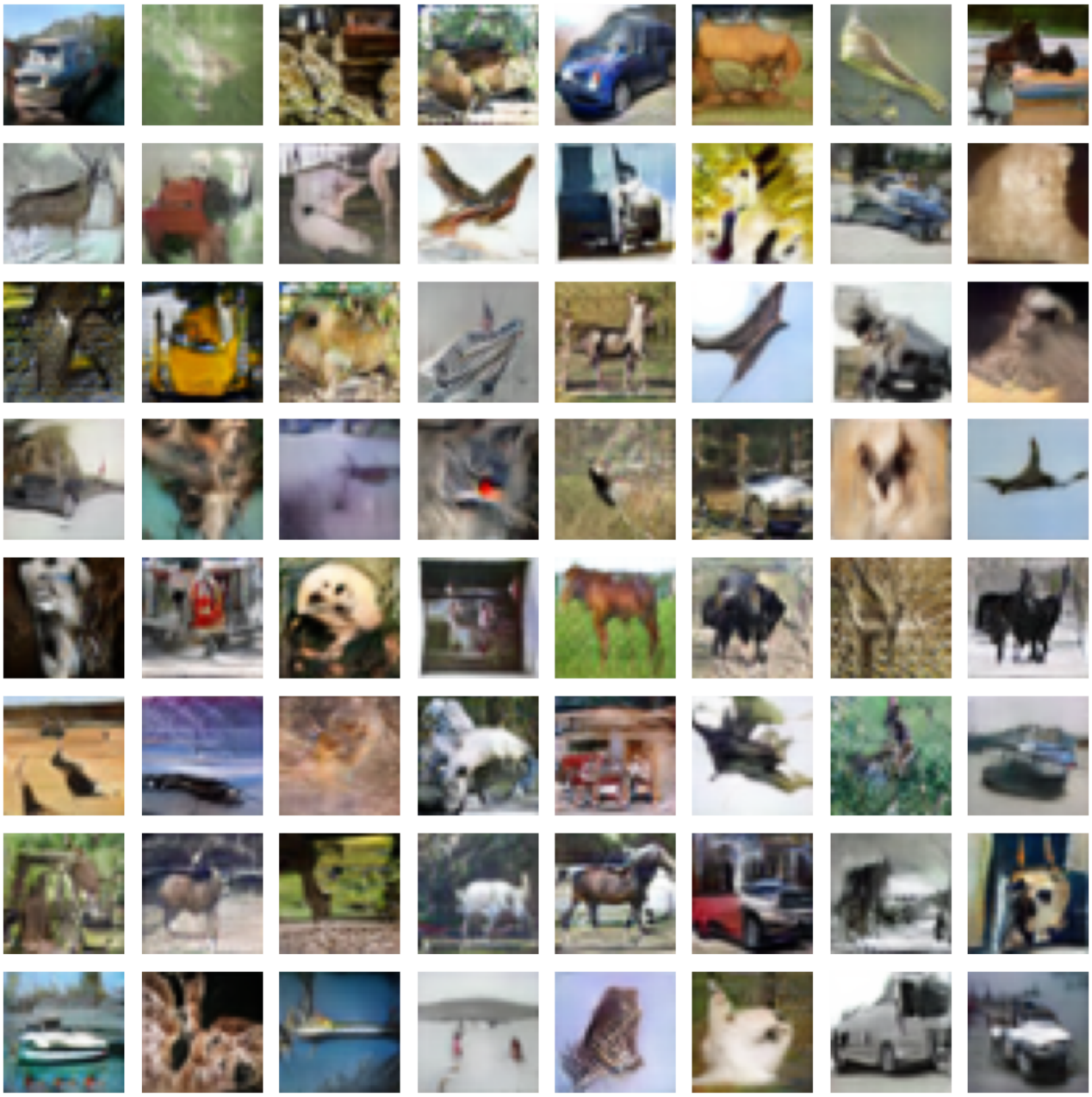

Figure 6: Generated samples of our Wasserstein GAN trained on CIFAR-10

References

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2017. URL `http://proceedings.mlr.press/v70/arjovsky17a.html`.

A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. *7th International Conference on Learning Representations, ICLR 2019*, 2019.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

I. J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. 2016. URL `http://arxiv.org/abs/1701.00160`.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 1998.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

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