Deep Learning

Generative Adversarial Networks and Cycle-GAN

ENSAE 2020/2021

Mastère Spécialisé - Data Science

Ban VAN TRUONG- Bertrand VUILLEMOT

Table des matières

1 Introduction

Generative adversarial networks (GANs) introduced in 2014 by Ian J. Goodfellow and co-authors has quickly received a tremendous attention in the field of artificial intelligence. It is even considered as "the most interesting idea in the last 10 years in Machine Learning" by Facebook's AI research director, Yann LeCun. Despite its huge success and potential for real life applications, GANs is facing a number of common failure modes that ask for improvements. Indeed, GANs has been widely known to be slow and unstable to train. Consequently, all of these problems promptly become areas of active research. Particularly, one of the famous variants of GANs are Deep Convolution GANs (DCGANs) and Wasserstein GANs (WGANs) which have partly solved some of GANs problems. Besides, Cycle GANs is also an attractive technique that involves the automatic training of image-to-image translation models without paired examples.

2 Deep Convolution GANs

DCGANs was first described by Alec Radford, Luke Metz and Soumith Chintala in their paper titled "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" written in 2015. With an extensive model exploration, the authors identified the architecture of DCGANs as the most satisfied family of architectures that makes the training process more stable across a range of datasets and allows for training higher resolution an deeper generative models.

What actually makes DCGANs a better version of GANs ? DCGANs explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively. The core approach of stable DCGANs is described in five architecture guidelines as follows :

- 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.

The authors trained DCGANs on three datasets including Large-scale Scene Understanding (LSUN) (Yu et al., 2015), Imagenet-1k and a newly assembled Faces dataset. In the paper, the authors also give several tips about how to setup the optimizers, how to calculate the loss functions, and how to initialize the model weights (Details can be found in the original paper). Note that the output of the generator is fed through a tanh function to return it to the input data range of [-1,1]. It is worth noting the existence of the

batch norm functions after the conv-transpose layers, as this is a critical contribution of the DCGAN paper. These layers help with the flow of gradients during training. An image of the generator from the DCGAN paper is shown in Figure 1.

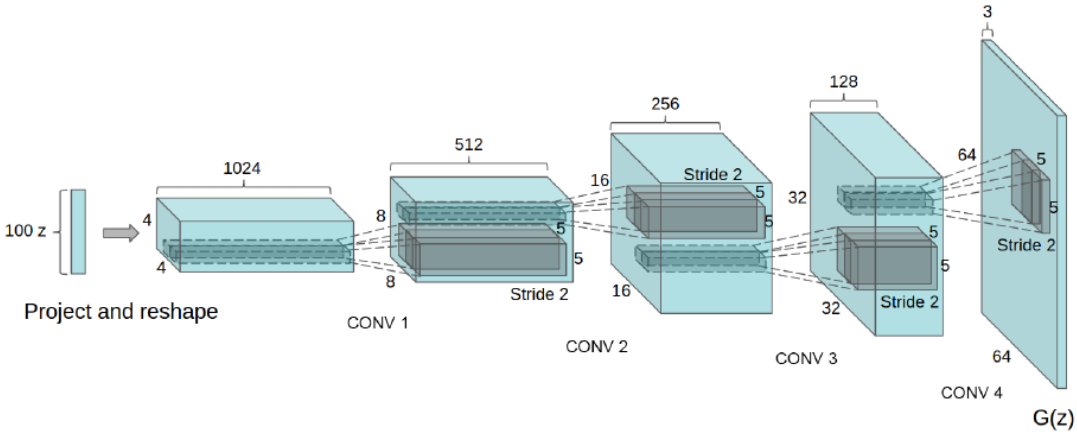


FIGURE 1 – DCGANs generator used for LSUN

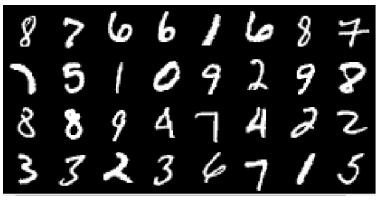
In the comparison between GANs and DCGANs, we implemented the two techniques with MNIST dataset. The result, shown in Figure 2, clearly proves the superior of DCGANs to GANs as DCGANs obtained a significant higher quality of images in only 126 epochs compared to 200 epochs of GANs tranning.

3 Wasserstein GANs

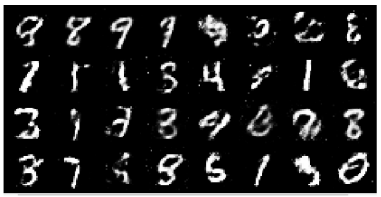
One of the popular problems of GANs is "mode collapse". More precisely, there are, in reality, cases that a generator produces a surprisingly plausible output, the generator then try to repeat only this satisfied output again and again. In response, the discriminator's best strategy is to learn to always reject that output. However, in the worst case that the next generation of discriminator are fooled, then as a result the generators rotate through solely a small set of output types. This has motivated Martin Arjovsky, Soumith Chintala and Léon Bottou to publish Wasserstein GAN in 2017, in which they prevent "mode collape" by trainning the discriminator to optimality without worrying about vanishing gradients. If the discriminator doesn't get stuck in local minima, it learns to reject the outputs that the generator stabilizes on.

The innovation of WGANs is changing from Jensen-Shannon (JS) divergence implemented by GANs to using the Earth-Mover (EM) distance or Wasserstein distance as follows :

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\Big[\|x - y\|\Big] \quad (1)$$

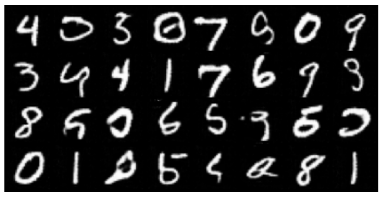


(a) Real MNIST images from 200 epochs of GANS



(b) Generated MNIST images from 200 epochs of GANS

(c) Real MNIST images from 126 epochs of DCGANS



(d) Generated MNIST from 126 epochs of DCGANS

FIGURE 2 – Images from DCGANs and GANs

Indeed, the EM distance is shown to be superior to all other compared distances that are total Variation (TV) distance, Kullback-Leibler (KL) divergence and, of course, JS divergence. Particularly, as demonstrated by a simple example, the authors shows that there exist sequences of distributions that don't converge under the JS, KL or TV divergence, but which do converge under the EM distance. Additionally, this example also shows that for the JS, KL and TV divergence, there are cases where the gradient is always 0. This is especially damning from an optimization perspective. Moreover, the already proved theorem 1 and 2 (details of proofs can be found in the original paper) have pointed out that when the supports are low dimensional manifolds in high dimensional space, it's very easy for the intersection to be measure zero and that continuity and differentiability of the loss function are guaranteed solely under EM distance. Lastly, it is proved that every distribution that converges under the KL, reverse-KL, TV, and JS divergences also converges under the Wasserstein divergence.

The next step is approximating the equation (1), however, estimating (1) is highly intractable. Thus, the authors replace (1) with the result of Kantorovich-Rubinstein duality in stead.

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}\Big[f(x)\Big] - \mathbb{E}_{x \sim \mathbb{P}_\theta}\Big[f(x)\Big] \quad (2)$$

where the supremum is over all 1-Lipschitz functions. However, the supremum (2) would become $K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta)$ if the supremum over K-Lipschitz functions is used in replacement of the supremum over 1-Lipschitz functions. Now together with the supposition of having a parameterized family of functions $\{f_w\}_{w \in W}$ that are all K-Lipschitz for some K, the theorem 3 (details of proof found in the original paper) gives us

$$\max_{w \in W} \mathbb{E}_{x \sim \mathbb{P}_r}\Big[f_w(x)\Big] - \mathbb{E}_{z \sim p(z)}\Big[f_w(g_\theta(z))\Big] \leq \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim \mathbb{P}_r}\Big[f(x)\Big] - \mathbb{E}_{x \sim \mathbb{P}_\theta}\Big[f(x)\Big] \tag{1}$$

$$= K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta).$$

Now, let's loop all this back to generative models. We want to train $P_\theta = g_\theta(Z)$ to match P_r. Intuitively, given a fixed g_θ, we can compute the optimal f_w for the Wasserstein distance. We can then backprop through $W(\mathbb{P}_r, \mathbb{P}_{g_\theta}(Z))$ to get the gradient for θ

$$\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = \nabla_\theta(\mathbb{E}_{x \sim \mathbb{P}_r}\big[f(x)\big] - \mathbb{E}_{z \sim Z}\big[f_w(g_\theta(z))\big]) \tag{2}$$

$$= -\mathbb{E}_{z \sim Z}\big[\nabla_\theta f_w(g_\theta(z))\big].$$

The full algorithm of WGAN process is described in Algorithm 1 below.

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

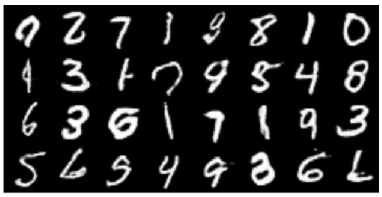
while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

The training process has now broken into three steps as follows :

- Step 1 : For a fixed θ, compute an approximation of $W(\mathbb{P}_r, \mathbb{P}_\theta)$by training f_w to convergence.

- Step 2 : Once we find the optimal f_w, compute the θ gradient $-\mathbb{E}_{z \sim Z}\big[\nabla_\theta f_w(g_\theta(z))\big]$ by sampling several $z \sim Z$.

- Step 3 : Update θ, and repeat the process.



(a) MNIST images from 100 epochs of DCGANs

(b) MNIST images from 500 epochs of DCGANs

FIGURE 3 – Images from different number of epochs of DCGANs

4 CycleGAN

4.1 DCGANs

Question 1 : Knowing that kernel size K = 5, stride S = 2 and the dimension of the input of discriminator is 32 x 32 x 3. Therefore, we have

$$(W - K + 2P) + 1 = (32 - 5 + 2P) + 1 = 32 \Leftrightarrow P = 2$$

which means that Padding = "Same" since P = S = 2.

Question 2 : Figure 3 presents the MNIST images generated by DCGANs for different number of epochs. It is clearly seen that the quality of images are significantly improved when increasing number of epochs.

4.2 CycleGAN

4.2.1 Question 1 :

With the number of iterations increasing, it is clear that the quality of numbers is also increasing as shown in the Figure 4.

4.2.2 Question 2 :

By choosing a different seed (i.e. 20), we can conclude that the choice of the seed is not producing noticeable changes to the outputs. In fact, since the seed affects the random noise only (the generator), the discriminators still penalize the "fake" outputs. It then forces the convergence towards pretty much the same outputs, even if the convergence speed may vary a little bit.

4.2.3 Question 3 :

Stating Zhu and al., "*with large enough capacity, a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an*

(a) USPS to MNIST (200^{th} iteration)

(b) USPS to MNIST (1200^{th} iteration)

(c) MNIST to USPS (200^{th} iteration)

(d) MNIST to USPS (1200^{th} iteration)

FIGURE 4 – Baseline CycleGAN model (**Seed = 4**, $\lambda = 0.015$)

(a) USPS to MNIST (200^{th} iteration)

(b) USPS to MNIST (1200^{th} iteration)

(c) MNIST to USPS (200^{th} iteration)

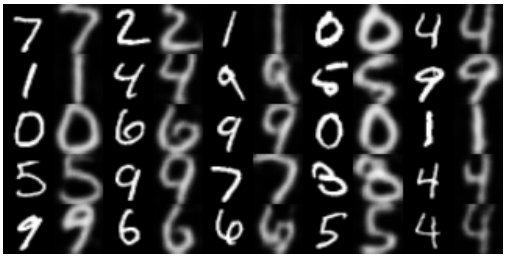
(d) MNIST to USPS (1200^{th} iteration)

FIGURE 5 – Changing seed CycleGAN model (**Seed = 20**, $\lambda = 0.015$)

(a) USPS to MNIST (200^{th} iteration)

(b) USPS to MNIST (1200^{th} iteration)

(c) MNIST to USPS (200^{th} iteration)



(d) MNIST to USPS (1200^{th} iteration)

FIGURE 6 – CycleGAN model–lowering consistency constraints ($\lambda = 0$)

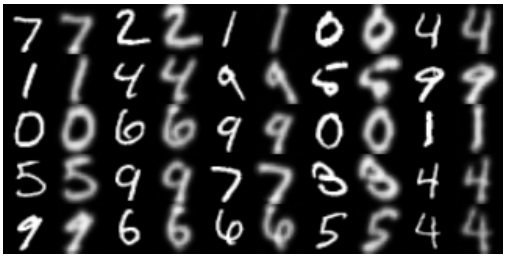
output distribution that matches the target distribution. Thus, adversarial losses alone cannot guarantee that the learned function can map an individual input x_i to a desired output y_i."

Without consistency constraints, the outputs' quality is clearly lower (Figure 6). Nonetheless, the quality is still increasing with higher number of iterations. This may be explained by the fact that $\lambda = 0$ makes the loss function (and then the objective function) depend only on adversarial loss $\mathcal{L}_{GAN}$. Thus the GAN is more unstable since we do not regularize both generators to be produce outputs consistent with inputs. Furthermore, increasing λ to 0.2 (multiplied by more than 10) is visually positive : numbers are much clearer in Figure 7.

(a) USPS to MNIST (200^{th} iteration)

(b) USPS to MNIST (1200^{th} iteration)

(c) MNIST to USPS (200^{th} iteration)



(d) MNIST to USPS (1200^{th} iteration)

FIGURE 7 – CycleGAN model–increasing consistency constraints ($\lambda = 0.2$)

5 Annexes

FIGURE 8 – EMOJI Baseline model : 10k iterations, seed = 4, $\lambda = 0.015$ – Samples at the 200^{th} iter and every thousands

FIGURE 9 – Changing seed=20

FIGURE 10 – Decreasing lambda ($\lambda = 0$)

FIGURE 11 – Increasing lambda ($\lambda = 0.05$)

6 References

[1] Alec Radford, Luke Metz and Soumith Chintala. 2016. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 1511.06434 arXiv

[2] Martin Arjovsky, Soumith Chintala, Léon Bottou. 2017. *Wasserstein GAN*. 1701.07875 arXiv

[3] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. 2020. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 1703.10593v7 arXiv

[4] hanwen0529. 2019. *https ://github.com/hanwen0529/Emoji-Translation-With-CycleGAN*

[5] Alexirpan. 2017. *https ://www.alexirpan.com/2017/02/22/wasserstein-gan.html*

[6] Aladdin Persson. *https ://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/GANs*