A Survey on GANs (Cycle GAN, WGAN, Style GAN)

Amirmahdi Ansaripour

Table of Contents

Cycle GAN

- It was first introduced in the “Unpaired Image-to-Image Translation Using Cycle Consistent Adversarial Networks” paper.
- As the name of the paper suggests, it is used for image-to-image translation; for example, between colored and grayscale images, edge-map and photograph, zebras and horses, etc.

Figure 1: Having two source domains (zebras, horses) or (Monet, photo), Cycle GAN can convert images from one domain to the other one.

Cycle GAN

- How does the model work?

It uses two GAN structures, in other words, two generators and two discriminators. The generators are shown as G and F.

Mathematically, they can be written as:

$$G: X \rightarrow Y$$
$$F: Y \rightarrow X$$

The discriminators, D_X and D_Y, try to distinguish between the transformed images and real images. Our objective is to optimize the components such that:

$$F\big(G(X)\big) \approx X$$
$$G\big(F(Y)\big) \approx Y \qquad (1)$$

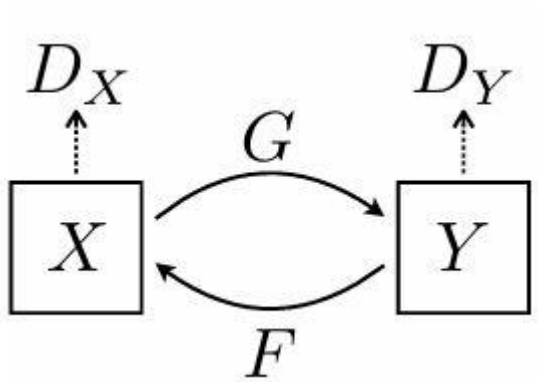

Figure 2: A scheme of Cycle GAN.

Cycle GAN

- Loss functions:

1. Typical loss functions of Vanilla GAN: The same is for F and D_X.

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))] \quad (2)$$

1. Cycle Consistency Loss: Norm-1 of the distance between $F(G(X))$ and X and $G(F(Y))$ and Y. Please refer to Eq. 1.

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]. \quad (3)$$

Cycle GAN

- The final loss function then becomes:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned} \tag{4}$$

Cycle GAN

- Cycle loss implementation:

```
# Cycle loss
map_to_B = generator_to_B(real_train_A, training = True)
cycled_A = generator_to_A(map_to_B, training = True)
map_to_A = generator_to_A(real_train_B, training = True)
cycled_B = generator_to_B(map_to_A, training = True)
total_cycle_loss = calc_cycle_loss(real_train_A, cycled_A) + calc_cycle_loss(real_train_B, cycled_B)
```

- Vanilla GAN losses implementation: Note that we apply each discriminator on each category of images (mapped and real), and there are two discriminators.

```
# Discriminator loss
disc_A_real = discriminator_on_A(real_train_A, training = True)
disc_A_generated = discriminator_on_A(map_to_A, training = True)
disc_B_real = discriminator_on_B(real_train_B, training = True)
disc_B_generated = discriminator_on_B(map_to_B, training = True)
disc_A_loss = calc_discriminator_loss(real = disc_A_real, generated = disc_A_generated)
disc_B_loss = calc_discriminator_loss(real = disc_B_real, generated = disc_B_generated)
# Generator loss
gen_A_loss = calc_generator_loss(generated = disc_A_generated)
gen_B_loss = calc_generator_loss(generated = disc_B_generated)
```

Cycle GAN

Some sample outputs (Zebra ↔ Horse domains).

Interesting point in this pair

Figure 3: Pairs of (source, transformed) images obtained in my implementation.

Table of Contents

Wasserstein GAN (WGAN)

- It was first introduced in the "**Wasserstein GAN**" paper, and then improved in the "**Improved Training of Wasserstein GANs**".
- It criticizes the KL[1] and JS[2] divergences, stating that they are not continuous and differentiable in all points. For example:

let $Z \sim U[0,1]$ and P_0 be the distribution of $(0,z) \in R^2$, a line from point $(0,0)$ to $(0,z)$.

Also, P_θ is the distribution of $(\theta,z) \in R^2$, a line from point $(\theta,0)$ to (θ,z).

$$JS(\mathbb{P}_0,\mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases} \qquad KL(\mathbb{P}_0\|\mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases} \tag{5}$$

1. Kullback-Leibler
2. Jensen-Shannon

WGAN

- As we can see, both JS and KL divergences are non-continuous,

so using them as objectives is not always beneficial.

- The paper proposes Wasserstein-1 distance, also called Earth-Mover's

distance, instead of previous divergences.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\| \right] , \quad (6)$$

In the above formula, inf means infimum or the greatest lower bound, and $\Pi(P_r, P_g)$ is the set of all joint distributions $\gamma(x, y)$.The Wasserstein loss between the P_0 and P_θ distributions is therefore:

$$W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta| \quad (7)$$

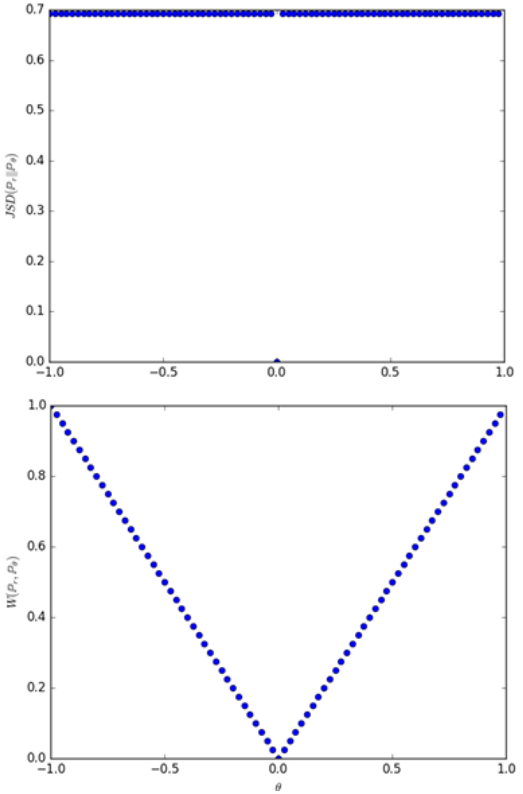

Figure 4: JS divergence (top) and Wasserstein loss (bottom) between P_0 and P_θ in the previous slide.

WGAN

- Intuitively, $W(P_r, P_g)$ states how much "mass" should be transported from x to y in order to transform the distribution P_r (real) into P_g (generated). WGAN tries to minimize this mass or at least find its greatest lower bound.
- However, finding the infimum in Eq. 6 is intractable, and the paper introduces solving a duality problem instead:

$$\max_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)] \qquad (8)$$

In Eq. 8, f is 1-Lipschitz function, which is estimated by a neural network:

$$\frac{|f(x) - f(y)|}{|x - y|} \leq 1 \qquad (9)$$

WGAN

- The final algorithm for training WGAN
- Two points:

1. The discriminator in Vanilla GAN is named critic, whose duty is to estimate f with weights w.

2. In each epoch, first, the critic is trained in n_{critic} inner loops, and then the generator is trained.

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.

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

Figure 5: Training algorithm of WGAN with simple clipping.

3. Line 7 enforces f to be 1-Lipschitz. The paper discusses that c is a hyper-parameter and in practice should take values bigger than 1. It highlights the disadvantages of setting c to very low or high values.

WGAN

- Enforcing f to be 1-Lipschitz through clipping is a very naïve approach. In a second paper named “Improved Training of Wasserstein GANs”, a penalty term is introduced when training the critic (Line 7).

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

1: **while** θ has not converged **do**
2: **for** $t = 1, ..., n_{\text{critic}}$ **do**
3: **for** $i = 1, ..., m$ **do**
4: Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
5: $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
6: $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
7: $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
8: **end for**
9: $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
10: **end for**
11: Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
12: $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
13: **end while**

Figure 6: Training algorithm of WGAN with gradient penalty term method.

WGAN

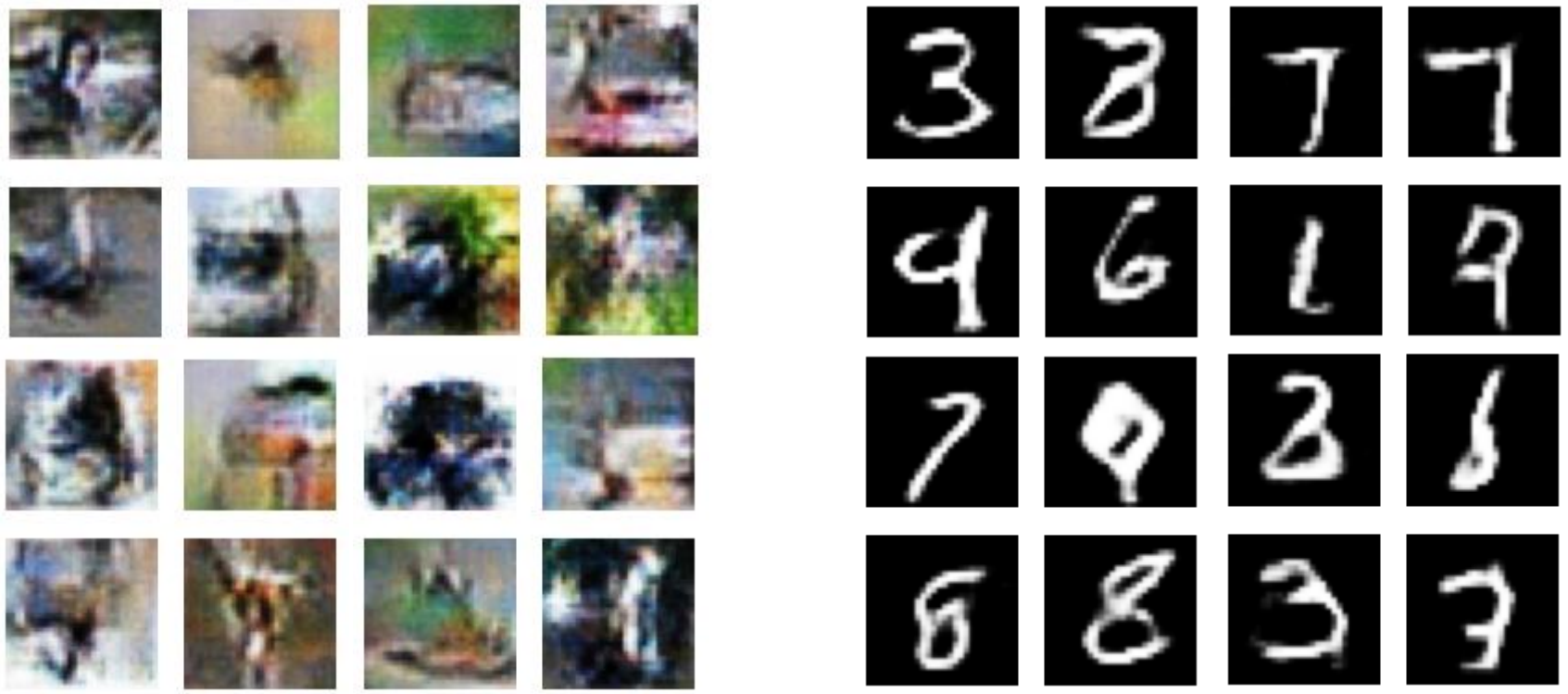

Figure 7: My output generated samples on CIFAR-10 (left) and MNIST (right) datasets.

Table of Contents

Style GAN

References

[1] Zhu, J.Y., Park, T., Isola, P. and Efros, A.A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2223-2232).

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

[3] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A.C., 2017. Improved training of wasserstein gans. *Advances in neural information processing systems*, *30*.