GAN collections

Zhi Li

June 12, 2019

1 GAN

1.1 Applications

GAN is kind-of unsupervised learning for generating sequences, graphs, videos etc. Check out all the applications here. collections of awesome GAN applications. Gernerally, input some random noise, the model will be trying to output the data satisfying to your domain. Some interesting examples like horse to zebra, shoes transformation, and face aging animation.

Problems lies in the aera that how to transform the random input towards something meaningful. This corresponds to the data distribution transformation, which will later display in the loss function.

The philosophy behind GAN is that, we train two frameworks, one for generator which input random noise and output useful results, and the other is discriminator which will drive generator to produce good results. This framework corresponds to a minimax two-player game. Generator will be forcing to generate something to fool discriminator while discriminator will classify the fake data.

Some related work to do so including deep-Boltzmann machine, Deep Belief Networks, Variational AutoEncoder (VAE)

1.2 Adversarial Nets

In his paper, he suggests that optimal Discriminator D is achieved when G is fixed.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 1: Algorithm description

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{1}$$

Theorem. If $p_g = p_{data}$ which means the global minimum. C(G) will achieve the value of $-log4$.

2 Various GAN

2.1 MMGAN

The one proposed by Goodfellow, the original loss function used for discriminator is $-log(1-D(x))$ (cross-entropy)

2.2 NSGAN

In NSGAN, the difference between it with MMGAN is that it starts to use $-log(D(x))$ because it proves that, at the begining, the gradient for $-log(1 - D(x))$ is small which may not reasonable as we need the gradient to be large to converge. So with $log(D(x))$, it initialized a higher gradient at first and decreasing progressively.

2.3 f-GAN

First, it proves that every divergence can be represented by $D_f(P \mid Q) = \int_x q(x) f(\frac{p(x)}{q(x)})$, and f function should comply two criteria: 1. convex function, 2. $f(1) = 0$. When $f(x) = xlog(x)$, it is KL divergence, and if $f(x) = -log(x)$, it is reversed KL-divergence.

Fenchel Conjugate, it is saying that every f-divergence has a conjugate function which could also be optimized towards our objective. $f^*(t) = max(xt - f(x))$

A collection of f-divergence functions can be found here: f-GAN: Training generative neural samples using variational divergence minimization

In conclusion, choosing different f-divergence seems affect the results little.

3 CGAN

3.1 Applications

3.2 Diagram

3.3 Loss Function

4 WGAN

4.1 Loss Function

In WGAN, it tries to minimize the so called **Earth Movement** divergence. imagine two distribution P and Q, now if we expect to move distribution P towards Q, then how much effort should we put? This is exactly what we seek for minimizing.

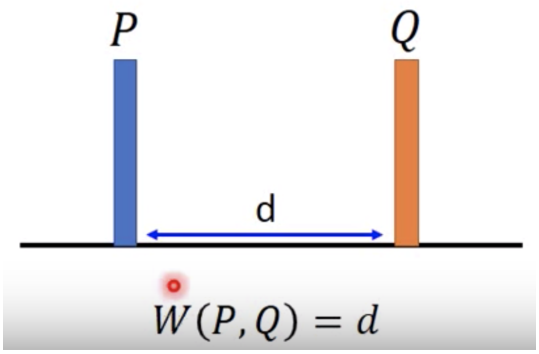

Figure 2: two distribution P adn Q, the distance between P and Q is d. W-GAN is performing to minimize this d

According to the loss function, we should minimize the following equation:

$$W(P_{data}, P_G) = \max_{D \in 1-Lipschitz} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\} \tag{2}$$

But, when applying gradient descent optimizer, it looks infeasible to force weights optimize with constraints "1-Lipschitz". Hence, the author proposed a method called "weight clipping". Put it simply, if weight larger than a threshold, then cut it becoming thaw-like.

4.1.1 Generator loss

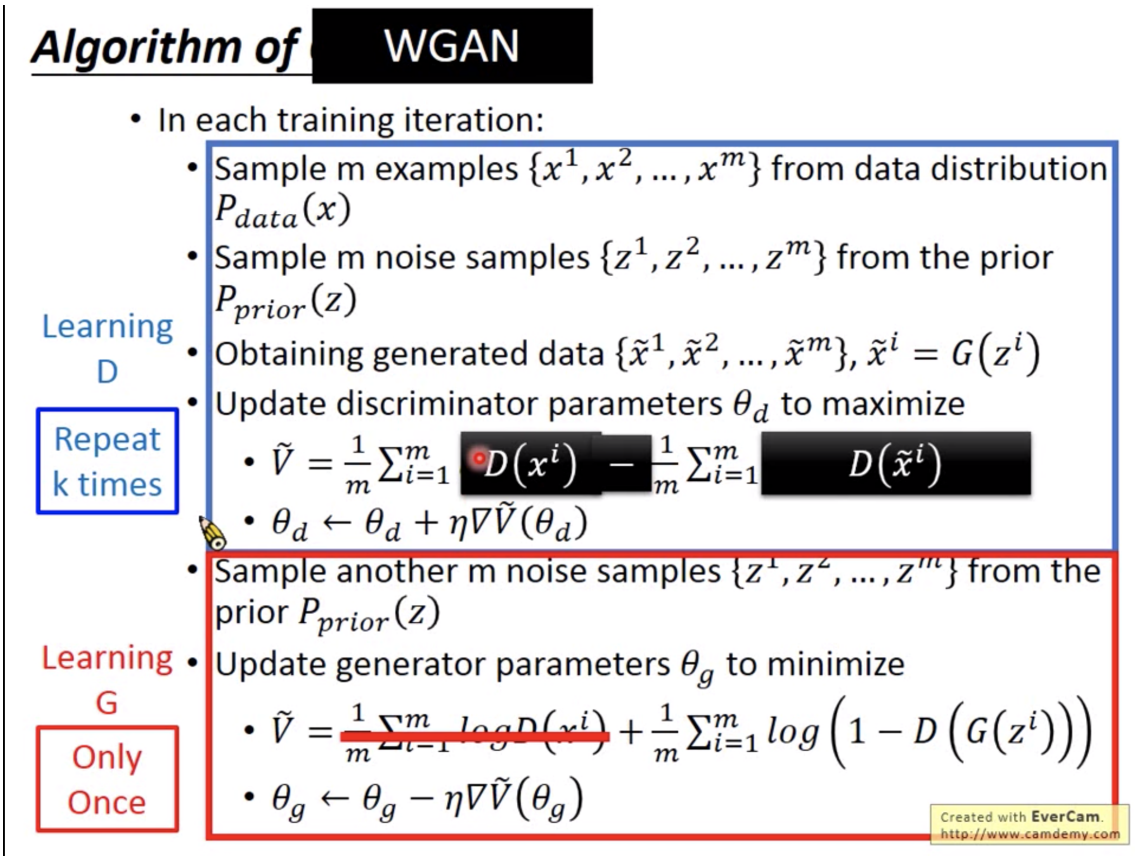

Figure 3: Description about WGAN

5 improved WGAN

5.1 improvement compared to WGAN

A differentiable 1-Lipschitz function if and only if it has gradient with norm less than or equal to 1 everywhere.

$$W(P_{data}, P_G) = \max_{D}\{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)] - \lambda \sum_{x} \max(0, | \bigtriangledown_x D(x) | -1)dx\} \quad | \bigtriangledown_x D(x) | \leq 1 \tag{3}$$

it leverages an additional term "regularization" to satisfy the constrain that WGAN has proposed. But it is impossible to sample all x from that function. Instead, author decides to sample x from a penalty distribution. Then the equation becomes

$$W(P_{data}, P_G) = \max_{D}\{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)] - \lambda E_{x\sim Penalty}[max(0, | \bigtriangledown_x D(x) | -1)]\} \tag{4}$$

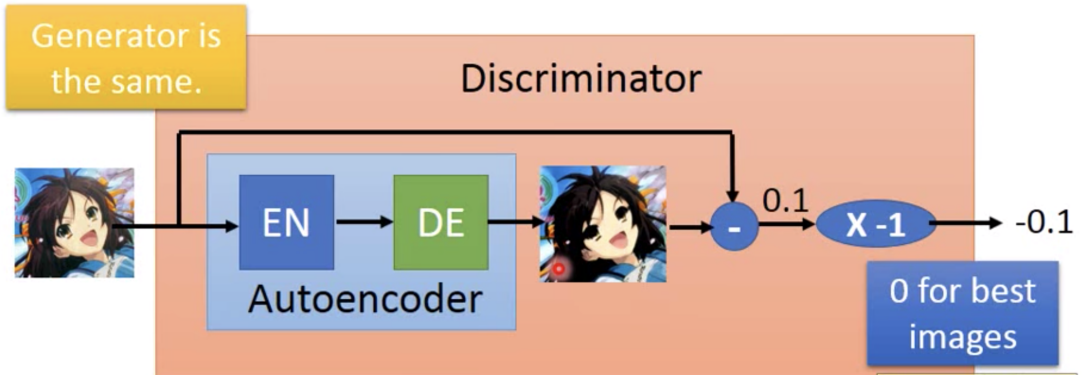

Figure 4: discriminator in EBGAN

6 Energy-based GAN

6.1 difference

- using auto-encoder as discriminator D
  basically, it is saying that if an image feeding into discriminator, returns small reconstruction error, and then it proves this image is good.

7 Divide and Conquer

7.1 Patch GAN

Patch GAN is basically train the image by dividing it into sub-patches

7.2 Stack GAN

Stack GAN is used to generate high resolution image by creating multi-stages discriminator and generator. In the original paper, it intakes embedding texts and then in the first stage, generate 64x64 dimension images. After that, it creates 256x256 dimension in the second generator and discriminator.

7.3 Progressive Growing GAN

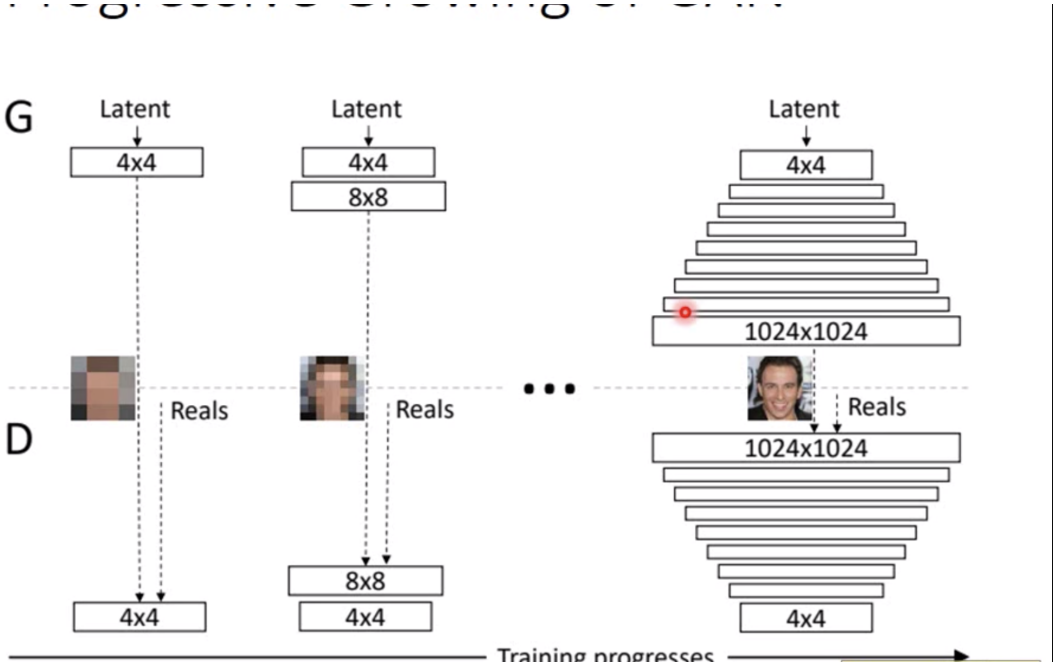

Figure 5: Progressive growing GAN

8 SGAN

8.1 Applications

8.2 Diagram

8.3 Loss Function

8.3.1 Generator loss

9 infoGAN

9.1 Applications

infoGAN could be used to control the image you want to generate. For instance, you can specify the input code to determine the digit to be bold/italic.

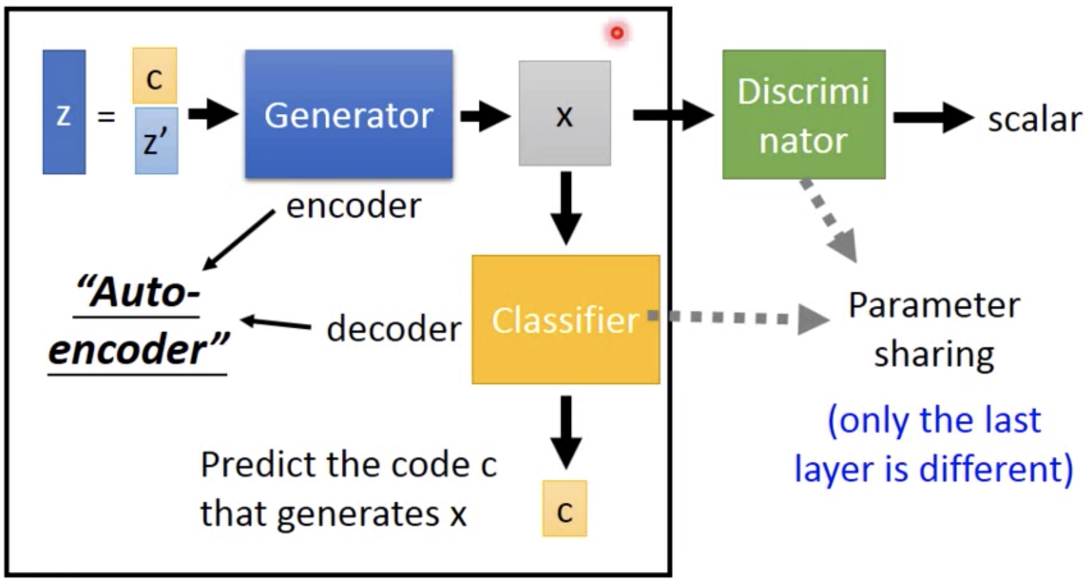

Figure 6: infoGAN

9.2 Diagram

10 Feature Disentangle

10.1 Applications

Feature disentangle is used to generate the series variations for the object. For instance, we want to generate the face with different projections. That is to say, if you understand the feature within the code, then you can extract that feature, to generate something meaningful.

11 Couple GAN

11.1 Applications

Style Transfer

11.2 diagram

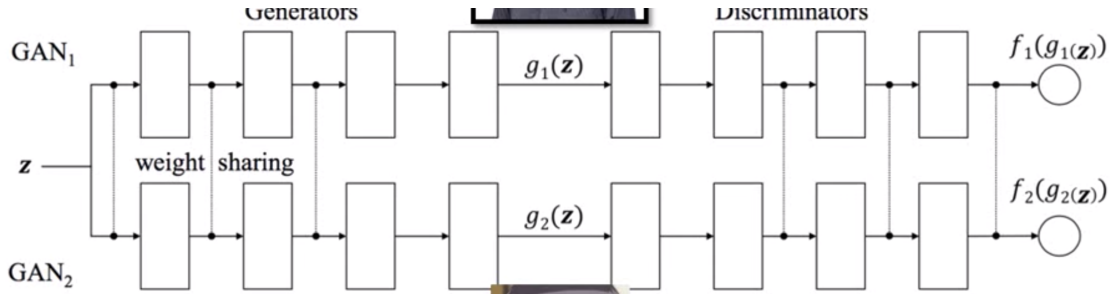

Figure 7: CoGAN

12 UNIT: Unsupervised Image-to-image Translation

12.1 Applications

image transfer, day-to-night,attribute based image translation, cat-to-cheetah.github site

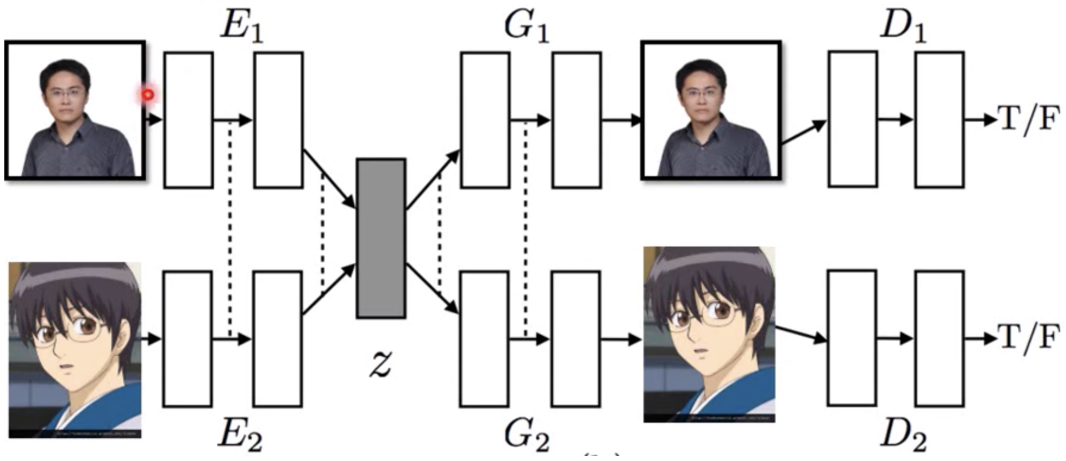

Figure 8: UNIT

13 tempoGAN

tempoGAN: A Temporally Coherent, Volumetric GAN for Super-resolution Fluid Flow

13.1 Applications

It is one of the methods that have been coupling with traditional physics. In this paper, the author uses tempoGAN to generate super-resolution results from traditional fuilds simulation. Traditionally, 2D or 3D fluids model is constrained by the grid size. course grid may result in the numerical instability while fine grids of course wastes more time and may not feasible considering the equipment. But with the tempoGAN, we are able to generate high resolution fluids with less efforts.

13.2 Key Points

- first work to synthesize four-dimensional physics fields with neural networks
- temporal discriminator
- infer high resolution results with physical properties like velocity and vorticities.
- Physical-aware data augmentation

13.3 Diagram

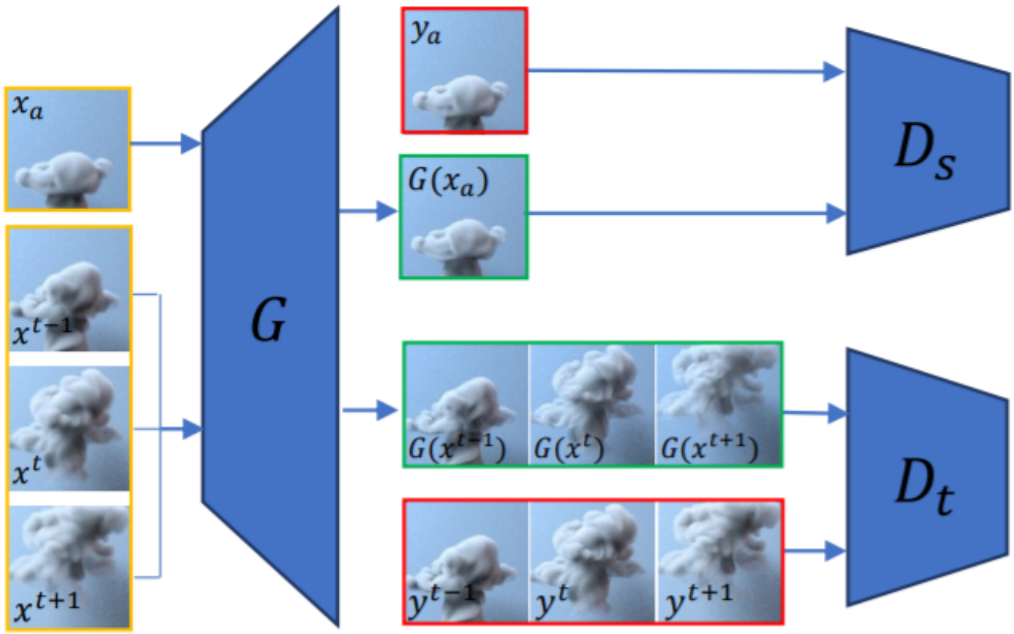

Figure 9: schematic representation of tempoGAN framework

13.4 Loss Function

13.4.1 Generator loss

14 DehazeGAN

14.1 Key Points

- U-Net model with Tiramisu
- Perceptual loss (Vgg)
- Patch-GAN for discriminator

14.2 Diagram

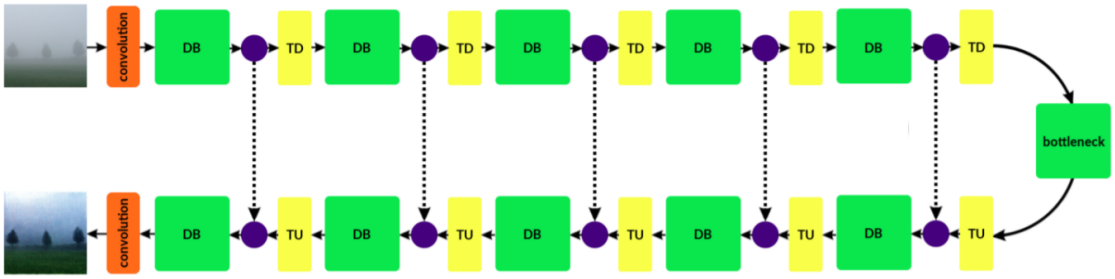

Figure 10: schematic representation of dehazeGAN model framework

A simple U-Net framework is utilized in this context, sharing information between encoder and decoder networks. The generator replaces U-Net with Tiramisu and each layer contains (BatchNorm-ReLU-Convolution) operations.

As for discriminator, it uses the same discriminator network (Patch GAN) as used in the original conditional GAN paper. It performs patch-wise comparison of the target and generated images, rather than pixel-wise comparison.

14.3 Loss function

$$Loss_{cGAN} = E_{(x,y)}[log(D(x,y))] + E_{x,z}[log(1 - D(x, G(x,z)))] \tag{5}$$

$$Loss_{total} = W_{gan} * L_{Adv} + W_{L1} * L_{L1} + W_{vgg} * L_{vgg} \tag{6}$$

Perceptual Loss

$$L_{vgg} = \frac{1}{CWH} \sum_{c=1}^{C} \sum_{w=1}^{W} \sum_{h=1}^{H} ||V(G(x,z)^{c,w,h}) - V(y^{c,w,h})|| \tag{7}$$

where, V represents the output channels, width and height respectively.

It says that L2 or smooth L1 loss would improve the image quality. L2 Loss is more susceptible to artefacts than smooth L1 loss. using Feature Reconstruction Perceptual Loss provides a much visually pleasing result.

15 Summary

GANS	Archive	Loss Function	
GAN	Arxiv	$L_D^{GAN} = E[log(D(x))] + E[log(1 - D(G(z))] L_G^{GAN} = E[log(D(G(z)]$	