

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH**



BÁO CÁO ĐỒ ÁN CUỐI KỲ

GENERATIVE ADVERSARIAL NETWORKS

**CS115.M13.KHCL
THÀNH VIÊN THỰC HIỆN
20521553 Võ Hoàng Lộc
20520821 Võ Minh Trí
19521724 Huỳnh Anh Kiệt**

**GIẢNG VIÊN HƯỚNG DẪN
TS. LƯƠNG NGỌC HOÀNG**

TP. HỒ CHÍ MINH, 12/2021

Mục lục

1	Generative Adversarial Network.....	1
1.1	Định nghĩa	1
1.2	Kiến trúc của một mô hình GAN.....	1
1.2.1	Discriminator.....	1
1.2.2	Generator	1
2	Toán cho GAN.....	2
2.1	Thuật toán huấn luyện GAN.....	2
2.2	Hàm loss	3
2.2.1	Discriminator loss function.....	3
2.2.2	Generator loss function	3
2.2.3	Binary Cross Entropy	3
2.3	Tối ưu hóa mô hình.....	4
2.3.1	Huấn luyện discriminator.....	4
2.3.2	Huấn luyện generator.....	5
3	Các mô hình GAN	8
3.1	1DGAN.....	8
3.2	Deep Convolutional GAN	8
3.3	Mô hình GAN sử dụng hàm loss khác.....	9
3.3.1	Wasserstein GAN.....	9
3.3.2	Least Squares GAN.....	10
3.4	Conditional GAN.....	11

Phần 1 Generative Adversarial Network

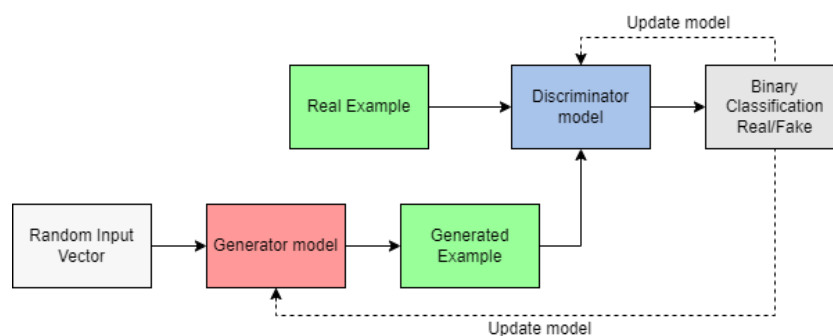
1.1 Giới thiệu về GAN

Generative Adversarial Network – GAN, là một generative model được giới thiệu lần đầu bởi Ian Goodfellow qua paper "Generative Adversarial Nets" năm 2014.

GAN là một mô hình học không giám sát nhưng được huấn luyện như một mô hình học có giám sát với hai thành phần là generator và discriminator.

Sau một quá trình học, GAN có thể tự sinh ra một khuôn mặt mới, một con người, một đoạn văn, chữ viết, bản nhạc giao hưởng,...

1.2 Kiến trúc của một mô hình GAN



Hình 1.1. Kiến trúc mô hình GAN

1.2.1 Discriminator

Discriminator là mô hình có khả năng phân loại các mẫu dữ liệu khác nhau. Đầu vào là mẫu dữ liệu thực từ tập dữ liệu và mẫu dữ liệu giả được tạo bởi generator, đầu ra là kết quả phân loại.

Mục tiêu sau quá trình học là nâng cao khả năng phân loại chính xác nhiều nhất có thể, từ đó báo kết quả cho generator để generator có thể tạo dữ liệu tốt hơn.

1.2.2 Generator

Generator là mô hình có khả năng tạo ra mẫu dữ liệu mới từ random vector (noise) các biến từ latent space (không gian các biến được sinh ngẫu nhiên từ phân phối Gauss).

Mục tiêu sau quá trình học là cải thiện khả năng tạo dữ liệu để đánh lừa discriminator nhiều nhất có thể, khiến discriminator phân loại dữ liệu của mình là thực.

Phần 2

Toán cho GAN

Kí hiệu:

x : dữ liệu thực (real data)

z : noise (input của generator)

$G(z)$: dữ liệu được tạo ra từ generator

$D(x)$: đánh giá của discriminator đối với dữ liệu thật

$D(G(x))$: đánh giá của discriminator với dữ liệu được tạo ra

$Error(a, b)$: độ lỗi giữa a và b

E : giá trị kì vọng (expectation)

$V(a, b)$: hàm giá trị của a và b (value function)

$p(x)$: phân phối xác suất của x

2.1 Thuật toán huấn luyện GAN

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

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

end for

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

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

end for

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

Hình 1.2. Thuật toán huấn luyện GAN, từ "Generative Adversarial Nets"

Tóm tắt thuật toán:

Với mỗi lần lặp trong n lần huấn luyện mô hình:

- Lặp lại k lần việc cập nhật mô hình discriminator bằng cách tăng dần stochastic gradient của nó.
- Sau đó cập nhật 1 lần generator bằng cách giảm dần stochastic gradient của nó.

2.2 Hàm loss

Trong paper gốc của Ian Goodfellow năm 2014, hàm loss của mô hình GAN được định nghĩa như một minimax objective function:

$$\min_G \max_D V(G, D) = \min_G \max_D \{\log(D(x)) + \log(1 - D(G(z)))\} \quad (2.1)$$

Là sự kết hợp của hai hàm:

1. Gradient **ascent** trên discriminator

$$\max_D \{\log(D(x)) + \log(1 - D(G(z)))\} \quad (2.2)$$

2. Gradient **descent** trên generator

$$\min_G \{1 - \log(D(G(z)))\} = \max_G \{\log(D(G(z)))\} \quad (2.3)$$

Trong thực tế, ta nên chia ra xử lý từng hàm loss riêng biệt sẽ dễ dàng hơn.

2.2.1 Discriminator loss function

Mục tiêu của discriminator là phân biệt chính xác dữ liệu thật là đúng và dữ liệu được tạo ra là sai. Xem đúng bằng 1 sai bằng 0 \Rightarrow ta có thể định nghĩa hàm loss của discriminator như sau:

$$L_D = \text{Error}(D(x), 1) + \text{Error}(D(G(z)), 0) \quad (2.4)$$

Hiểu đơn giản: nếu $D(G(z))$ cho ra kết quả đánh giá là $G(z)$ giống thật 90% ($D(G(z)) = 0.9$) \Rightarrow độ lỗi sẽ bằng giá trị dương hiệu của $D(G(z))$ với 0: $\text{Error} = 0.9 - 0 = 0.9$

2.2.2 Generator loss function

Ngược lại với discriminator, mục tiêu của generator là khiến discriminator dán nhãn cho dữ liệu được tạo ra là đúng \Rightarrow định nghĩa hàm loss của generator:

$$L_G = \text{Error}(D(G(z)), 1) \quad (2.5)$$

2.2.3 Binary Cross Entropy

Discriminator là một bài toán phân loại nhị phân, nên ta có thể sử dụng Binary Cross Entropy – BCE, làm hàm mất mát. Công thức Cross Entropy:

$$H(p, q) = E_{x \sim p(x)} [-\log q(x)] \quad (2.6)$$

Trong bài toán phân loại, các biến ngẫu nhiên là rời rạc, nên kì vọng có thể được thể hiện dưới dạng một tổng:

$$H(p, q) = - \sum_{x \in \mathcal{X}} p(x) \log q(x) \quad (2.7)$$

Trong trường hợp chỉ có hai nhãn cần phân loại (0 và 1), Cross Entropy có thể được đơn giản hóa thành Binary Cross Entropy:

$$H(y, \hat{y}) = - \sum (y \log \hat{y} + (1 - y) \log(1 - \hat{y})) \quad (2.8)$$

Hàm Error được sử dụng trên phương trình (2.4) và (2.5) là một bài toán Binary Cross Entropy \Rightarrow thay Error bằng công thức, ta được:

$$L_D = - \sum_{x \in \mathcal{X}, z \in \mathcal{Z}} (\log(D(x)) + \log(1 - D(G(z)))) \quad (2.9)$$

$$L_G = - \sum_{z \in \mathcal{Z}} \log(D(G(z))) \quad (2.10)$$

2.3 Tối ưu hóa mô hình

Tối ưu hóa mô hình GAN là hành động tìm các tham số cho generator và discriminator sao cho tối ưu được hàm loss, đồng nghĩa với việc huấn luyện mô hình.

Huấn luyện GAN là huấn luyện hai mô hình generator và discriminator một cách luân phiên: khi huấn luyện discriminator thì ta cập nhật giá trị D, giữ giá trị G cố định và ngược lại.

2.3.1 Huấn luyện discriminator

Xem lại hàm loss của G và D được mô tả trong paper năm 2014 của Ian Goodfellow:

$$V(G, D) = E_{x \sim p_{data}} [\log(D(x))] + E_{z \sim p_z} [\log(1 - D(G(z)))] \quad (2.11)$$

Trong thực tế, ta quan tâm đến phân phối dữ liệu được tạo ra bởi generator (p_g) hơn phân phối xác suất của noise (p_z), gọi $y = G(z)$, thay y và p_g vào $V(G, D)$, ta được:

$$\begin{aligned} V(G, D) &= E_{x \sim p_{data}} [\log(D(x))] + E_{y \sim p_g} [\log(1 - D(y))] \\ &= \int_{x \in \mathcal{X}} p_{data}(x) \log(D(x)) dx + \int_{y \in \mathcal{Y}} p_g(y) \log(1 - D(y)) dy \\ &= \int_{x \in \mathcal{X}} (p_{data}(x) \log(D(x)) + p_g(x) \log(1 - D(x))) dx \end{aligned} \quad (2.12)$$

Đạo hàm riêng của $V(G, D)$ với $D(x)$:

$$\frac{\partial V(G, D)}{\partial D(x)} = \frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)} \quad (2.13)$$

Mục tiêu của việc huấn luyện discriminator là tối đa hóa $V(G,D)$. Với đạo hàm, ta biết rằng $V(G,D)$ đạt cực đại khi:

$$\frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1-D(x)} = 0 \quad (2.14)$$

Sắp xếp lại phương trình (2.14), ta được giá trị $D(x)$ là discriminator tối ưu, kí hiệu $D^*(x)$:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \quad (2.15)$$

Nếu x là dữ liệu thực, ta sẽ cần $p_{data}(x) \approx 1$ và $p_g(x) \rightarrow 0 \Rightarrow D^*(x) \approx 1$, discriminator sẽ phân loại x là thực.

Mặt khác, nếu $x = G(z)$, tức x là dữ liệu được tạo ra, ta cần $p_{data}(x) \approx 0 \Rightarrow D^*(x) \approx 0$, discriminator sẽ phân loại x là mẫu giả.

2.3.2 Huấn luyện generator

Để huấn luyện generator, ta giữ cố định discriminator (cố định giá trị D). Thế $D^*(x)$ mà ta xác định ở phương trình (2.15) vào hàm giá trị V , ta được:

$$\begin{aligned} V(G, D^*) &= E_{x \sim p_{data}} [\log(D^*(x))] + E_{x \sim p_g} [\log(1 - D^*(x))] \\ &= E_{x \sim p_{data}} \left[\log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right] + E_{x \sim p_g} \left[\log \frac{p_g(x)}{p_{data}(x) + p_g(x)} \right] \\ &= \int_{x \in \mathcal{X}} p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} + p_g(x) \log \frac{p_g(x)}{p_{data}(x) + p_g(x)} dx \\ &= \int_{x \in \mathcal{X}} (\log 2 - \log 2) p_{data}(x) + p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \\ &\quad + (\log 2 - \log 2) p_g(x) + p_g(x) \log \frac{p_g(x)}{p_{data}(x) + p_g(x)} dx \\ &= -\log 2 \int_{x \in \mathcal{X}} p_{data}(x) + p_g(x) dx + \int_{x \in \mathcal{X}} p_{data}(x) \left(\log 2 + \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) \\ &\quad + p_g(x) \left(\log 2 + \log \frac{p_g(x)}{p_{data}(x) + p_g(x)} \right) dx \\ &= -\log 2.2 + \int_{x \in \mathcal{X}} p_{data}(x) \log \frac{2 \cdot p_{data}(x)}{p_{data}(x) + p_g(x)} + p_g(x) \log \frac{2 \cdot p_g(x)}{p_{data}(x) + p_g(x)} dx \\ &= -\log 4 + \int_{x \in \mathcal{X}} p_{data}(x) \log \frac{p_{data}(x)}{\frac{p_{data}(x) + p_g(x)}{2}} + p_g(x) \log \frac{p_g(x)}{\frac{p_{data}(x) + p_g(x)}{2}} dx \end{aligned} \quad (2.16)$$

Mục tiêu của việc huấn luyện generator là tối thiểu hóa $V(G, D)$. Với phương trình (2.16), ta có thể thấy $V(G, D^*)$ nhỏ nhất bằng khi tích phân bằng 0 \Leftrightarrow các logarit bằng 0 $\Leftrightarrow p_g(x) = p_{data}(x)$.

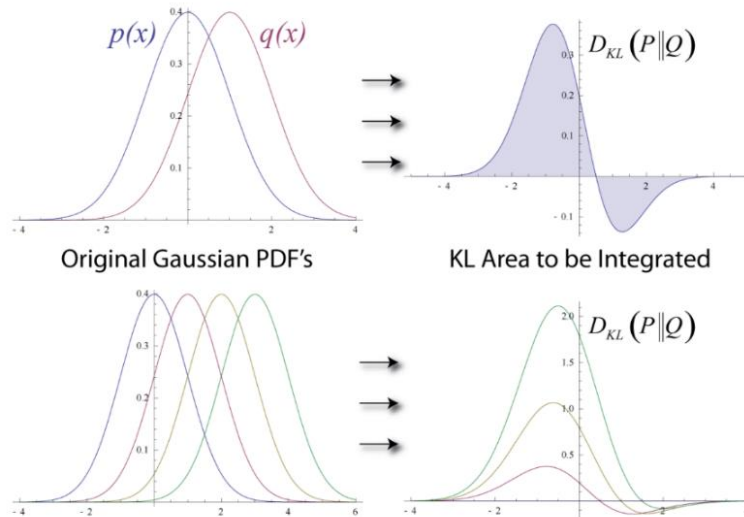
Tuy nhiên trong bài báo gốc Ian Goodfellow không dùng cách tính toán trên mà sử dụng phân kỳ Jensen-Shannon.

Phân kỳ Kullback-Leibler (KL divergence - D_{KL}), còn gọi là entropy tương đối, là một khoảng cách thống kê cho ta biết phân phối xác suất Q xấp xỉ P như thế nào.

Gọi P và Q là hai phân phối xác suất của một biến ngẫu nhiên rời rạc x. $\forall x \in \mathcal{X}: P(x) + Q(x) = 1 \wedge P(x) > 0 \wedge Q(x) > 0$, $D_{KL}(P(x) \parallel Q(x))$ được định nghĩa:

$$D_{KL}(P \parallel Q) = \int_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} \quad (2.18)$$

- Tính chất:
- Phân kỳ KL không âm: $D_{KL}(P \parallel Q) \geq 0$, dấu bằng xảy ra khi $P = Q$
 - Phân kỳ KL không đối xứng: $D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$



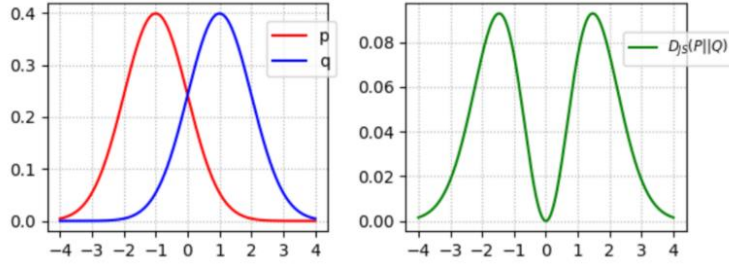
Hình 2.1. Phân kỳ Kullback-Leibler và sự bất đối xứng, từ Wikipedia

Phân kỳ Jensen-Shannon (JS divergence - D_{JS}) là một phiên bản đối xứng và mượt hơn của phân kỳ Kullback-Leibler, được định nghĩa:

$$D_{JS}(P \parallel Q) = \frac{1}{2} D_{KL}(P \parallel M) + \frac{1}{2} D_{KL}(Q \parallel M) \quad (2.19)$$

với $M = \frac{1}{2}(P + Q)$.

- Tính chất:
- Khi logarit được tính theo cơ số 2: $0 \leq D_{JS}(P \parallel Q) \leq 1$
 - Phân kỳ JS đối xứng, bằng 0 khi $P = Q$.



Hình 2.2. Phân kỳ Jensen-Shannon và sự đối xứng

Trở lại với bài toán tối thiểu hóa $V(G, D^*)$:

Ta thể biểu diễn phương trình (2.16) bằng phân kỳ KL như phương trình (2.18):

$$V(G, D^*) = -\log 4 + D_{KL}(p_{data} \parallel \frac{p_{data} + p_g}{2}) + D_{KL}(p_g \parallel \frac{p_{data} + p_g}{2}) \quad (2.20)$$

Tiếp tục biểu diễn phương trình (2.20) bằng phân kỳ JS như phương trình (2.19):

$$V(G, D^*) = -\log 4 + 2 \cdot D_{JS}(p_{data} \parallel p_g) \quad (2.21)$$

Mục tiêu của việc huấn luyện generator là tối thiểu hóa $V(G, D)$. Với phương trình (20), ta có thể thấy $V(G, D^*)$ nhỏ nhất bằng $-\log 4$ khi $D_{JS}(p_{data} \parallel p_g)$ nhỏ nhất bằng 0.

Mà $D_{JS}(P \parallel Q) = 0 \Leftrightarrow P = Q$, có nghĩa việc ta cần là để p_{data} và p_g càng gần bằng nhau càng tốt.

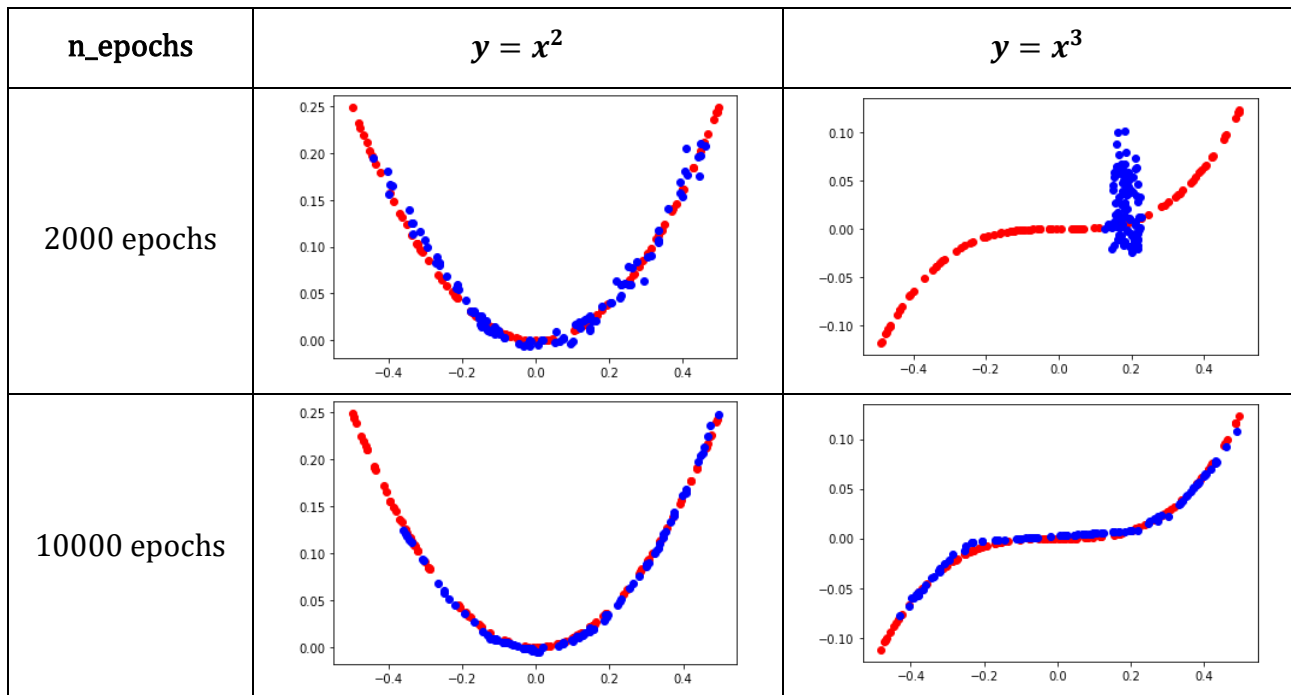
Đó cũng là thứ ta cần generator học được, p_g càng gần p_{data} tức dữ liệu generator tạo ra càng giống dữ liệu thật.

Phần 3

Các mô hình GAN

3.1 Original GAN

Huấn luyện GAN truyền thống để tạo hình ảnh đồ thị của hàm số 1 biến $y = f(x)$ và đây là kết quả sau 2000 và 10000 lần huấn luyện:



Các điểm màu đỏ là dữ liệu thực của hàm số, các điểm màu xanh là dữ liệu được tạo ra bởi generator.

3.2 Deep Convolutional GAN

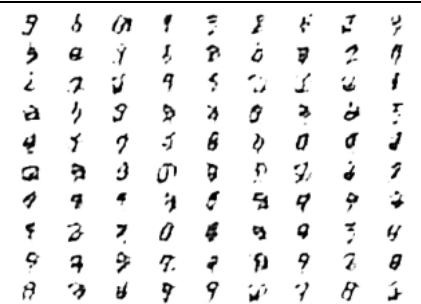
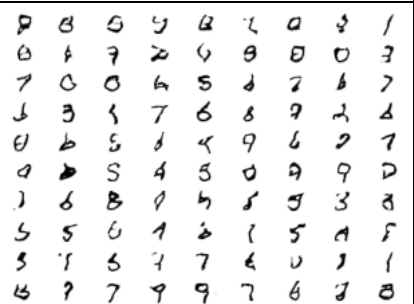

DCGAN là mô hình GAN cơ bản sử dụng mạng neural tích chập (Convolutional Neural Network - CNN) như generator và discriminator để làm việc với các dữ liệu hình ảnh.

Đặc điểm của DCGAN:

1. Generator là các neural được xếp theo phân đoạn, sử dụng ReLu activation cho tất cả các lớp, trừ lớp output sử dụng Tanh activation
2. Discriminator là các neural được xếp theo chuỗi, sử dụng LeakyRelu activation cho tất cả các lớp
3. Normalization được sử dụng hàng loạt trong 2 mô hình
4. Sử dụng một kiến trúc sâu hơn thay vì chỉ các lớp ẩn được kết nối đầy đủ

DCGAN có điểm hạn chế ở hàm loss BCE. discriminator sau khi huấn luyện sẽ đưa thông tin cho generator học nhưng generator lại không học được, làm cho giá trị gradient descent của nó thường bị triệt tiêu.

Có thể thấy rõ trong kết quả huấn luyện trên bộ dữ liệu MNIST, khác biệt sau 10 và 40 epoch là rất lớn nhưng từ 40 đến 100 epoch lại không cải thiện quá nhiều.

10 epochs:	40 epoch:	100 epoch:
		

Để khắc phục hạn chế này có thể thay hàm loss BCE bằng hàm loss giúp mô hình ổn định hơn.

3.3 Mô hình GAN sử dụng hàm loss khác

3.3.1 Wasserstein GAN

WGAN sử dụng khoảng cách Wasserstein, hay khoảng cách Earth Mover, làm hàm loss thay cho Binary Cross Entropy để tìm ra phân phối gần nhất giữa p_g và p_{data} .

Khoảng cách Wasserstein là giá trị cận dưới lớn nhất (infimum) của khoảng cách giữa phép dịch chuyển p_{data} sang p_g :

$$W(p_{data}, p_g) = \inf_{\gamma \in \Pi(p_{data}, p_g)} E_{(x,y) \sim \gamma} [\|x - y\|] \quad (3.1)$$

$\Pi(p_{data}, p_g)$ là phân phối đồng thời của p_{data} và p_g , với p_{data} và p_g là phân phối biên.

Việc giải trực tiếp bài toán khoảng cách Wasserstein là khá khó nên ta sử dụng phép đối ngẫu Kantorovich-Rubinstein để đơn giản hóa:

$$W(p_{data}, p_g) = \sup_{\|f\|_L \leq 1} \{E_{x \sim p_{data}}[f(x)] - E_{x \sim p_g}[f(x)]\} \quad (3.2)$$

Trong đó sup là cận trên nhỏ nhất (supremum) và f là hàm số 1-Lipschitz tuân theo ràng buộc $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$.

Khác biệt giữa GAN và WGAN:

1. Discriminator không còn phân loại thật, giả nữa mà sẽ trở thành Critic để đánh giá dữ liệu được tạo ra từ -1 đến 1 điểm. Điểm càng cao sẽ càng giống thật.
2. Thay lớp output của Critic từ sigmoid thành linear projection.
3. Critic được huấn luyện nhiều hơn Generator trong cùng 1 epoch.
4. Sử dụng RMSProp để cập nhật gradient descent với momentum = 0

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.

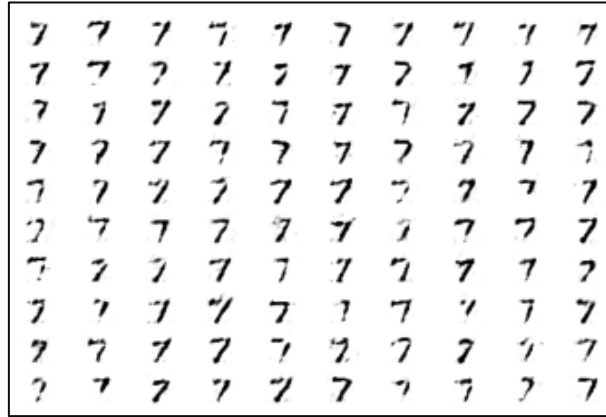
```

1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  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:      $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ 
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:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ 
12: end while

```

Hình 3.3. Thuật toán huấn luyện WGAN, từ "Wasserstein GAN"

Kết quả khi huấn luyện WGAN trên bộ dữ liệu MNIST:



Hình 3.3. 100 chữ số 7 được tạo bởi WGAN (10 epochs)

3.3.2 Least Squares GAN

LSGAN sử dụng Mean Squared Error, hay L2 Loss làm hàm loss.

Công thức Mean Squared Error:

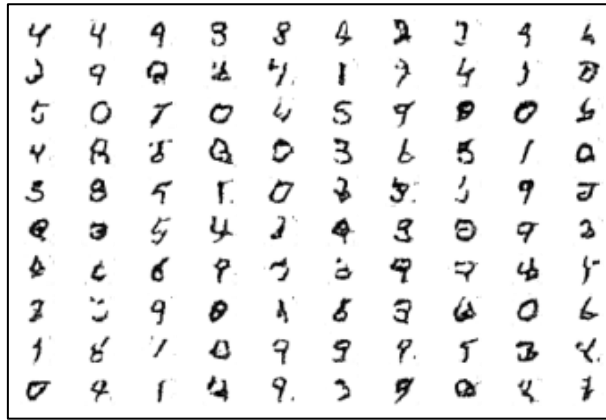
$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3.3)$$

Discriminator loss và generator loss của LSGAN:

$$\min_D V_{LSGAN}(D) = \frac{1}{2} E_{x \sim p_{data}(x)} (D(x) - b)^2 + \frac{1}{2} E_{z \sim p_{data}(z)} (D(G(z)) - a)^2 \quad (3.4)$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2} E_{z \sim p_z(z)} (D(G(z)) - c)^2 \quad (3.5)$$

Với a, b lần lượt là nhãn cho dữ liệu giả, thực, c là giá trị mà G muốn D tin tưởng cho dữ liệu giả.



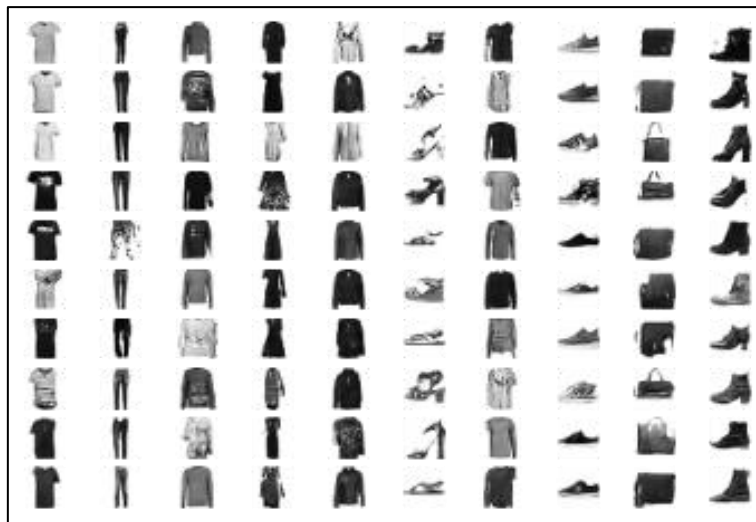
Hình 3.4. 100 chữ số viết tay được tạo với LSGAN (20 epochs)

3.4 Conditional GAN

CGAN là mô hình GAN cơ bản có thông tin bổ sung được thêm vào generator và discriminator. Thông tin ấy có thể là bất cứ thứ gì như nhãn phân lớp, tên, cấp độ,...

Nếu thông tin bổ sung là nhãn phân lớp, generator sẽ tạo dữ liệu mới thuộc đúng lớp. Discriminator sẽ đánh giá xem dữ liệu được tạo ra có giống dữ liệu thực thuộc lớp đó hay không.

Dưới đây là kết quả khi huấn luyện CGAN với bộ dữ liệu Fashion-MNIST để tạo hình ảnh với nhãn:



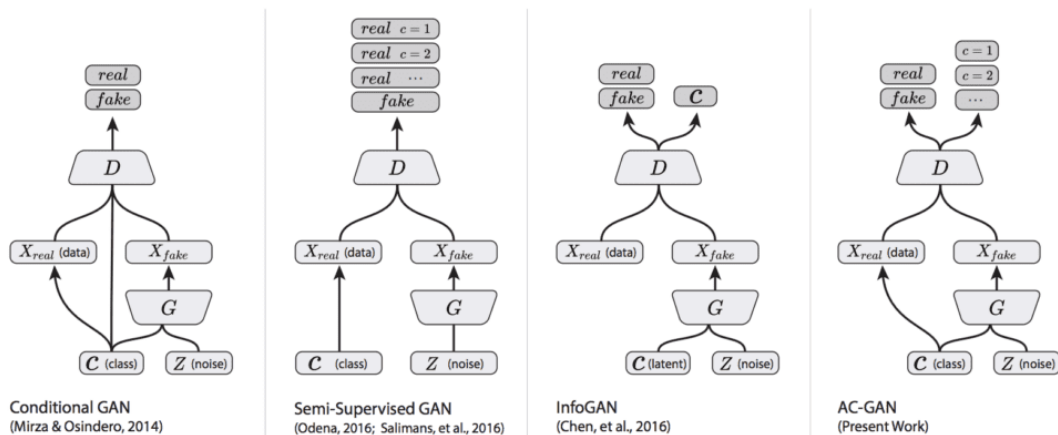
Hình 3.5. Hình ảnh quần áo được tạo và phân lớp bởi CGAN (100 epochs)

Các loại Conditional GAN:

1. **Information Maximizing GAN – InfoGAN**, là mô hình có thể tách rời các thuộc tính của hình ảnh được tạo. Ví dụ với khuôn mặt, các thuộc tính kiểm soát là hình dạng của khuôn mặt, màu tóc, kiểu tóc,...
2. **Auxiliary Classifier GAN – AC-GAN**, mô hình có discriminator có thể đánh giá thật giả cộng với việc dự đoán nhãn của hình ảnh được tạo.
3. **Semi-supervised – SGAN**, bộ dữ liệu truyền vào discriminator sẽ có nhiều mẫu không được gắn nhãn, ít mẫu được gắn nhãn, discriminator sẽ kết hợp cả:
 - Học không giám sát trên các mẫu không gắn nhãn để phân biệt thật, giả như GAN

truyền thống.

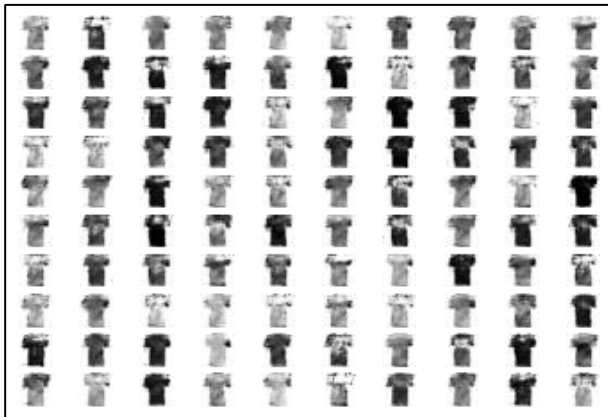
- Học có giám sát trên các mẫu được gắn nhãn như ACGAN để dự đoán nhãn lớp.



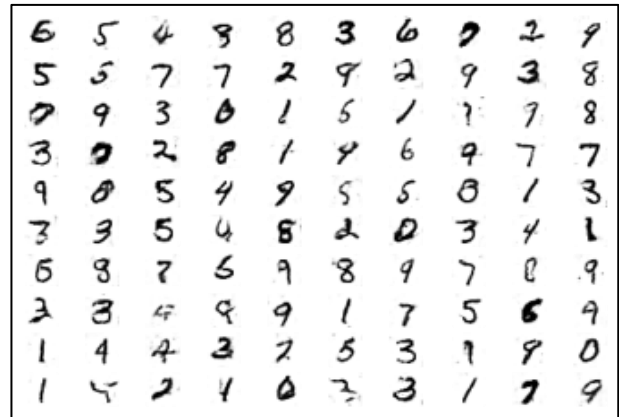
Hình 3.6 Tóm tắt 4 loại mô hình Conditional GAN, từ "Conditional Image Synthesis With Auxiliary Classifier GANs".

Một số ứng dụng của CGAN như:

- Cải thiện chất lượng ảnh (Image Super-Resolution):** sinh ra phiên bản ảnh có độ phân giải cao hơn ảnh gốc.
- Sáng tạo nghệ thuật (Creating Art):** vẽ tranh, viết kịch và sinh ảnh nghệ thuật.
- Thay đổi nội dung ảnh (Image-to-Image Translation):** thay đổi nội dung bức ảnh từ ban ngày thành ban đêm, mùa hè thành mùa xuân,...



Hình 3.6. Kết quả huấn luyện ACGAN tạo T-shirt bằng Fashion-MINST(100 epochs)



Hình 3.7. Tạo chữ viết tay bằng SGAN (20 epochs)

Tài liệu tham khảo

- [1] Ian J. Goodfellow & others (2014). Generative Adversarial Nets. Computer Science and Operational Research Department, University of Montreal.
- [2] Yang Wang (2020). A Mathematical Introduction To Generative Adversarial Nets (Gan).
- [3] Alec Radford, Luke Metz, Soumith Chintala (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.
- [4] Martin Arjovsky , Soumith Chintala & Léon Bottou (2017). Wasserstein GAN.
- [7] Xi Chen & others (2016). InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets.
- [8] Xudong Mao & others (2017). Least Squares Generative Adversarial Networks.
- [9] Mehdi Mirza & Simon Osindero (2014), Conditional Generative Adversarial Nets.
- [10] Jason Brownlee (2019). Generative Adversarial Networks with Python.
- [11] Wikipedia. Kullback-Leibler divergence. https://en.wikipedia.org/wiki/Kullback-Leibler_divergence, truy cập ngày 23/12/2021
- [12] Wikipedia. Jensen-Shannon divergence. https://en.wikipedia.org/wiki/Jensen-Shannon_divergence, truy cập ngày 23/12/2021