**Đồ án môn học**

**THỊ GIÁC MÁY TÍNH**

Đề tài : SINH ẢNH MẠNG GAN

**Hà Nội , 11/2022**

Lời mở đầu

Một trong những xu hướng nghiên cứu thu hút được đông đảo các nhà khoa học, có tính ứng dụng cao và phát triển mạnh mẽ trong những năm gần đây trong Deep Learning có lẽ là GAN. Chính vì thế bài viết này mình sẽ giới thiệu về GAN, các ứng dụng, kiến trúc thuật toán và phương pháp huấn luyện theo hướng tiếp cận đơn giản nhất, dựa trên kiến trúc GAN đầu tiên được Ian GoodFellow giới thiệu vào năm 2014, để dễ hiểu hơn cho người bắt đầu. Các kiến trúc GAN nâng cao hơn sẽ được mình trình bày ở các bài sau. Những kiến trúc này vẫn dựa trên ý tưởng chủ đạo của model GAN đầu tiên nhưng có sự cải tiến đầu vào, phương pháp huấn luyện, hàm loss function một chút để kết quả học được tốt hơn.

Xin chân thành cảm ơn!

MỤC LỤC

[Chương 1. TỔNG QUAN 5](#_Toc121473076)

[1. Các thuật ngữ 5](#_Toc121473077)

[2, Phân biệt Discriminative và Generative model 5](#_Toc121473078)

[3. Tính xác suất 6](#_Toc121473079)

[Chương 2. MẠNG GAN 7](#_Toc121473080)

[1. Các dạng generative model 7](#_Toc121473081)

[1.2, Mô hình ẩn (implicit model) 8](#_Toc121473082)

[2. GAN model 9](#_Toc121473083)

[3. Nguyên lý hoạt động của GAN 9](#_Toc121473084)

[3.1, Generator 10](#_Toc121473085)

[3.2, Discriminator 10](#_Toc121473086)

[3.3, Hàm loss function 11](#_Toc121473087)

[3.4, Quá trình huấn luyện 12](#_Toc121473088)

[Chương 3. THỰC NGHIỆM 13](#_Toc121473089)

[1. Tập dữ liệu 13](#_Toc121473090)

[2. Kiến trúc mạng GAN 13](#_Toc121473091)

[3. Code 13](#_Toc121473092)

[Chương 4. KẾT LUẬN 18](#_Toc121473093)

# Chương 1. TỔNG QUAN

GAN thuộc nhóm generative model. Generative là tính từ nghĩa là khả năng sinh ra, model nghĩa là mô hình. Vậy hiểu đơn giản generative model nghĩa là mô hình có khả năng sinh ra dữ liệu. Hay nói cách khác, GAN là mô hình có khả năng sinh ra dữ liệu mới.

GAN viết tắt cho Generative Adversarial Networks. Generative giống như ở trên, Network có nghĩa là mạng (mô hình), còn Adversarial là đối nghịch. Tên gọi như vậy là do GAN được cấu thành từ 2 mạng gọi là Generator và Discriminator, luôn đối nghịch đầu với nhau trong quá trình train mạng GAN.

## 1. Các thuật ngữ

* GAN: Model GAN, là lớp mô hình có khả năng tạo ra những dữ liệu giống với thật nhất.
* Discriminative: Mô hình phân biệt, có tác dụng phân loại.
* Generative: Mô hình sinh, nhằm tạo ra mẫu dữ liệu dựa trên nhãn đã biết.
* Explicit model: Mô hình hiện, một dạng của Generative model sử dụng các hàm phân phối xác suất lý thuyết để sinh mẫu dữ liệu.
* Implicit model: Mô hình ẩn, một dạng khác của Generative model, không sử dụng phân phối xác suất lý thuyết mà thay vào đó, mẫu sinh được sinh ra từ mô hình. GAN là một lớp mô hình như vậy.
* Zero-sum game: Một dạng trong lý thuyết trò chơi khi lợi ích giữa 2 người chơi là xung đột.

Để làm quen với GAN model, chúng ta cần phải nắm vững hai lớp bài toán khác nhau trong Machine Learning đó là mô hình Generative và mô hình Descriminative.

## 2, Phân biệt Discriminative và Generative model

Diagram

Description automatically generated

Các mô hình Machine Learning có thể được phân chia thành lớp mô hình phân biệt (Discriminative) và mô hình sinh (Generative). Đây chỉ là một cách phân chia trong vô số các cách phân chia khác như: mô hình học có giám sát (supervised learning)/học không giám sát (unsupervised learning), mô hình tham số (parametric)/mô hình phi tham số (non parametric), mô hình đồ thị (graphic)/mô hình phi đồ thị (non-graphic),....Mô hình phân biệt sẽ dựa trên những biến đầu vào để dự báo nhãn hoặc giá trị . Về bản chất đây chính là một mô hình classification hoặc prediction. Mô hình sẽ dự báo đầu ra dựa trên các dấu hiệu là đầu vào đã biết và giá trị dự báo là một xác suất có điều kiện: . Trong đó là mục tiêu cần dự báo và được xem như điều kiện.

Một lớp mô hình khác làm nhiệm vụ trái ngược lại so với mô hình phân biệt là mô hình sinh. Tức là chúng ta sẽ cố gắng dự báo . Mô hình sẽ tập trung hơn vào việc tìm kiếm đặc trưng của dữ liệu như thế nào nếu như đã biết trước đầu ra của dữ liệu.  
Chúng ta sẽ dễ hình dung hơn thông qua một ví dụ dưới đây:  
Giả sử chúng ta có một bộ dữ liệu nợ xấu có biến đầu vào là và nhãn dự báo gồm hai trường hợp: 1 đối với Fraud và 0 đối với Non-Fraud .

Diagram

Description automatically generated

Mô hình phân biệt (bên trái hình vẽ) sẽ tìm cách xác định một đường biên phân loại để phân chia tốt nhất hai nhóm Fraud và Non-Fraud. Trái lại ở mô hình sinh ta đã biết trước dữ liệu thuộc nhóm Fraud hay Non Fraud . Dựa vào phân phối của khi nó rơi vào các trường hợp Fraud và Non Fraud, ta sẽ sinh ra một mẫu mới sát với thực tế nhất như chấm hình vuông trên hình. Lưu ý là với mô hình sinh chúng ta phải đã biết trước nhãn của dữ liệu.

## 3. Tính xác suất

Thông thường đối với Discriminative model để dự báo xác suất ta sẽ dựa vào một hàm số như sigmoid trong trường hợp hai nhãn hoặc softmax trong trường hợp nhiều hơn 2 nhãn. Chẳng hạn xác suất trong trường hợp hai nhãn:

Nhưng đối với Generative model để dự báo được xác suất ta phải dựa trên công thức bayes. Đây là công thức được sử dụng để tính xác suất có điều kiện mà mình đã nói qua tại công thức bayes. Bạn đọc sẽ dễ hình dung hơn qua ví dụ trực quan:  
Tính xác suất vỡ nợ (biến cố ) của một người có thu nhập dưới 10 triệu (biến cố ). Biết rằng:

Áp dụng công thức bayes để tính xác suất không quá khó:

Ví dụ ở trên là trường hợp đơn giản dữ liệu đầu vào một chiều, trong trường hợp dữ liệu đầu vào có nhiều chiều thì ta sẽ cần giả định các chiều là độc lập và qui về tích các xác suất của từng chiều .

Như vậy ta có thể nhận thấy xác suất của lớp mô hình sinh hoàn toàn được suy ra từ các phân phối xác suất của các chiều mà không dựa trên công thức hồi qui theo như mô hình phân biệt.

# Chương 2. MẠNG GAN

1. Các dạng generative model  
Có hai dạng chính của generative model đó là mô hình hiện (explicit model) và mô hình ẩn (implicit model). Chúng khác nhau ở những điểm sau:  
1.1, Mô hình hiện (explicit model)  
Có mục tiêu chính là tìm ra phân phối xác suất của dựa trên một hàm phân phối xác suất được giả định trước của đầu vào. Phân phối xác suất đó có thể là gaussian, T-student, Chi-Square, Fisher, poisson, bernoulli, phân phối nhị phân,.... Model sẽ tìm kiếm tham số phân phối phù hợp nhất với bộ dữ liệu thông qua hàm cực đại hợp lý (maximize likelihood function). Để sinh ra một dữ liệu mới, chúng ta chỉ cần lấy mẫu ngẫu nhiên từ phân phối xác suất đã được ước lượng.  
Các mô hình generative model tiêu biểu thuộc lớp mô hình hiện là:  
**Phương pháp cực đại hợp lý (maximum likelihood):**

* PPCA, Phân tích nhân tố (Factor analysis), Mô hình trộn lẫn (Mixture models).
* PixelCNN/PixelRNN
* Wavenet
* Mô hình tự hồi qui ngôn ngữ (Autoregressive language models).

**Phương pháp xấp xỉ cực đại hợp lý (approximate maximum likelihood):**

* Máy học Bolzmann (Bolzmann machines).
* Bộ mã hóa tự động biến đổi (Variational Autoencoder)

### 1.2, Mô hình ẩn (implicit model)

Chúng ta không cần sinh dữ liệu mới dựa trên giả định về hàm phân phối của dữ liệu. Thay vào đó, một bộ mô phỏng (simulator) hoặc chính xác là một mô hình có khả năng sinh ra dữ liệu giống với dữ liệu thật nhất thông qua quá trình huấn luyện hội tụ trên dữ liệu. Phương pháp này ước lượng tham số của phân phối xác suất tiền định và đồng thời cũng không sử dụng phương pháp cực đại hàm ước lượng hợp lý để mô phỏng phân phối. Dữ liệu sẽ được sinh ra trực tiếp từ mô hình. Tiêu biểu cho phương pháp này là:

• Mạng khớp moment (moment matching networks).

• GAN.

Chart, line chart

Description automatically generated

Mô tả sự khác biệt giữa mô hình ẩn (implicit model) bên trái và mô hình hiện (explicit model) bên phải. Dữ liệu của mô hình ẩn được sinh ra từ một mô hình mô phỏng. Trái lại dữ liệu từ mô hình hiện được rút ra từ phân phối xác suất mà phân phối xác suất này được học trên các điểm dữ liệu.

## 2. GAN model

GAN là lớp mô hình có mục tiêu là tạo ra dữ liệu giả giống với thật. GAN được viết tắt từ cụm từ *Generative Adversarial Networks* tức là một mạng sinh đối nghịch (Generative tương ứng với sinh và Adversarial là đối nghịch). Sở dĩ GAN có tên gọi như vậy là vì kiến trúc của nó bao gồm hai mạng có mục tiêu đối nghịch nhau đó là Generator và Descriminator.

## 3. Nguyên lý hoạt động của GAN

Cũng giống như các lớp model Discriminative và Generative đã trình bày ở mục 1. Generator và Descriminator có chức năng như sau:

A picture containing text, cat, screenshot

Description automatically generated

* **Generator**: Học cách sinh ra dữ liệu giả để lừa mô hình Discriminator. Để có thể đánh lừa được Discriminator thì đòi hỏi mô hình sinh ra output phải thực sự tốt. Do đó chất lượng ảnh phải càng như thật càng tốt.
* **Discriminator**: Học cách phân biệt giữa dữ liệu giả được sinh từ mô hình Generator với dữ liệu thật. Discriminator như một giáo viên chấm điểm cho Generator biết cách nó sinh dữ liệu đã đủ *tinh xảo* để qua mặt được Discriminator chưa và nếu chưa thì Generator cần tiếp tục phải học để tạo ra ảnh thật hơn. Đồng thời Discriminator cũng phải cải thiện khả năng phân biệt của mình vì chất lượng ảnh được tạo ra từ Generator càng ngày càng giống thật hơn. Thông qua quá trình huấn luyện thì cả Generator và Discriminator cùng cải thiện được khả năng của mình.

Generator và Discriminator tương tự như hai người chơi trong bài toán zero-sum game trong lý thuyết trò chơi. Ở trò chơi này thì hai người chơi xung đột lợi ích. Hay nói cách khác, thiệt hại của người này chính là lợi ích của người kia. Mô hình Generator tạo ra dữ liệu giả tốt hơn sẽ làm cho Discriminator phân biệt khó hơn và khi Discriminator phân biệt tốt hơn thì Generator cần phải tạo ra ảnh giống thật hơn để qua mặt Discriminator. Trong zero-sum game, người chơi sẽ có chiến lược riêng của mình, đối với Generator thì đó là sinh ra ảnh giống thật và Discriminator là phân loại ảnh thật/giả. Sau các bước ra quyết định của mỗi người chơi thì zero-sum game sẽ đạt được cân bằng Nash tại *điểm cân bằng* (Equilibrium Point).

### 3.1, Generator

Diagram

Description automatically generated

Sơ đồ kiến trúc của generator.

Generator về bản chất là một mô hình sinh nhận đầu vào là một tập hợp các véc tơ nhiễu **z** được khởi tạo ngẫu nhiên theo phân phối Gaussian. Ở một số lớp mô hình GAN tiên tiến hơn, input có thể làm một dữ liệu chẳng hạn như bức ảnh, đoạn văn bản hoặc đoạn âm thanh.

Từ tập véc tơ đầu vào **z** ngẫu nhiên, mô hình generator là một mạng học sâu có tác dụng biến đổi ra bức ảnh giả ở output. Bức ảnh giả này sẽ được sử dụng làm đầu vào cho kiến trúc Discriminator.

### 3.2, Discriminator

Timeline

Description automatically generated with medium confidence

Mô hình Discriminator sẽ có tác dụng phân biệt ảnh input là thật hay giả. Nhãn của mô hình sẽ là thậtnếu ảnh đầu vào của Discriminator được lấy tập mẫu huấn luyện và giả nếu được lấy từ output của mô hình Generator. Về bản chất đây là một bài toán phân loại nhị phân (binary classification) thông thường. Để tính phân phối xác suất cho output cho Discriminator chúng ta sử dụng hàm sigmoid.

### 3.3, Hàm loss function

Hàm loss function của model gan là một hàm kết hợp đồng thời giữa mục tiêu của Discriminator và mục tiêu của Generator.

Cơ chế hàm loss function gồm 2 phases của hàm loss function này:

Phase huấn luyện Descriminator: Mục tiêu của phase này là huấn luyện một mô hình Descriminator sao cho khả năng phân loại là tốt nhất. Ở phase này chúng ta hãy tạm thời coi không đổi và chỉ quan tâm đến vế . Đây là chính là đối của hàm cross entropy đối với trường hợp phân loại nhị phân. Thật vậy, chắc hẳn bạn còn nhớ mục tiêu của logistic regression đối với bài toán phân loại nhị phân là tối thiểu hóa một hàm cross entropy như sau:

Trong đó là xác suất dự báo nhãn từ mô hình logistic.  
Trong phương trình (1) thì cũng như . Hay nói cách khác đóng vai trò dự báo xác suất cho dữ liệu đầu vào. Chúng ta có hai khả năng xảy ra:

* Nếu đầu vào là ảnh thật thì và và do đó loss function tương ứng với ở phương trình (2). Giá trị này được coi như là ở phương trình (1). Kí hiệu ở phương trình (1) là phân phối xác suất của các điểm dữ liệu đầu vào, trong trường hợp ở phương trình (2) thì các quan sát có vai trò như nhau nên chúng có chung giá trị phân phối là .
* Trường hợp ảnh đầu vào là giả thì và . Khi đó đóng góp vào hàm loss function chỉ còn thành phần ở phương trình (2). Giá trị này được coi như là ở phương trình (1).  
  Đừng quên rằng chúng ta đảo dấu loss function để chuyến sang bài toán tìm max ở phương trình (1). Như vậy các bạn đã hiểu ý nghĩa của hàm phase tối ưu loss function cho Descriminator rồi chứ?

**Phase huấn luyện Generator:** Mục tiêu của phase này là củng cố khả năng tạo ảnh của Generator sao cho ảnh nó sinh ra là giống với thật nhất. Ở phase này ta coi như là không đổi và chỉ quan tâm đến sao cho giá trị dự báo xác suất từ đối với nó gần bằng 1 nhất, tức là ảnh giả được sinh ra giống ảnh thật nhất (xác suất càng gần 1 thì khả năng giống ảnh thật càng lớn). Như vậy sẽ càng lớn càng tốt. Đảo dấu của nó trong ta suy ra mục tiêu cần tối ưu là tối thiểu hóa .

### 3.4, Quá trình huấn luyện

Trong quá trình huấn luyện thì chúng ta sẽ kết hợp một cách xen kẽ giữa hai phase. batch đầu tiên chúng ta sẽ huấn luyện discriminator trước:

* Huấn luyện discriminator: Lấy mẫu một mini-batch kích thước là các nhiễu và là đầu vào của Generator. Đồng thời lấy mẫu một mini-batch khác kích thước là những điểm dữ liệu thật . Những dữ liệu này sẽ được sử dụng để cập nhật gradient descent theo phương pháp mini-batch gradient descent:

Do là huấn luyện trên mô hình Discriminator nên chỉ được cập nhật các hệ số trên mô hình Discrimator là . Các hệ số của Generator được đóng băng.

* Huấn luyện generator: Sau khi kết thúc batch huấn luyện trên discriminator chúng ta sẽ tiếp tục huấn luyện trên generator. một mini-batch kích thước được lựa chọn ra từ các nhiễu là được sử dụng như đầu vào huấn luyện. Gradient descent sẽ được tính trên dữ liệu này theo công thức:

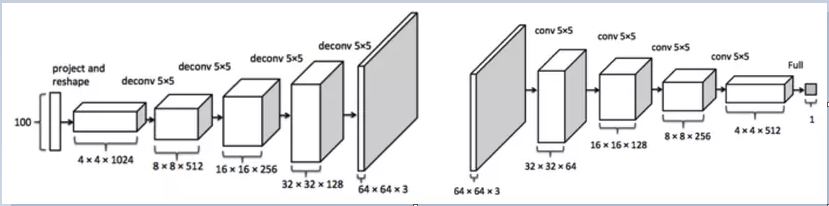
Lưu ý cập nhật gradient descent chỉ được áp dụng trên các hệ số của Generator là . Tiếp tục quá trình này cho tới khi tổng số lượt huấn luyện là đủ lớn hoặc loss của mô hình tiệm cận về 0.

# Chương 3. THỰC NGHIỆM

## 1. Tập dữ liệu

- Bộ chân dung các nhân vật Anime. Anime Dataset được public trên Kaggle theo link sau: [Kaggle anime-faces](https://www.kaggle.com/soumikrakshit/anime-faces)

## 2. Kiến trúc mạng GAN



## 3. Code

Import các thư viện cần thiết:

import numpy as np

import os

from PIL import Image

from matplotlib import pyplot as plt

import torch

import torchvision.transforms as T

from torch.utils.data import Dataset, DataLoader

from torchvision import datasets as torch\_dataset

from torchvision.utils import make\_grid

from torch import nn

from torch.nn import functional as F

Chuẩn hóa data

img\_size = 64

data\_dir = '../input/anime-faces/data'

data\_transforms = T.Compose([

T.Resize(img\_size),

T.CenterCrop(img\_size),

T.ToTensor(),

T.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),

])

anime\_dataset = torch\_dataset.ImageFolder(root=data\_dir, transform=data\_transforms)

dataloader = DataLoader(dataset=anime\_dataset, batch\_size=128, shuffle=True, num\_workers=4)

img\_batch = next(iter(dataloader))[0]

combine\_img = make\_grid(img\_batch[:32], normalize=True, padding=2).permute(1,2,0)

plt.figure(figsize=(15,15))

plt.imshow(combine\_img)

plt.show()



Xây dựng kiến trúc mạng GAN

def weights\_init(m):

classname = m.\_\_class\_\_.\_\_name\_\_

if classname.find('Conv') != -1:

nn.init.normal\_(m.weight.data, 0.0, 0.02)

elif classname.find('BatchNorm') != -1:

nn.init.normal\_(m.weight.data, 1.0, 0.02)

nn.init.constant\_(m.bias.data, 0)

def Conv(n\_input, n\_output, k\_size=4, stride=2, padding=0, bn=False):

return nn.Sequential(

nn.Conv2d(

n\_input, n\_output,

kernel\_size=k\_size,

stride=stride,

padding=padding, bias=False),

nn.BatchNorm2d(n\_output),

nn.LeakyReLU(0.2, inplace=True),

nn.Dropout(p=0.2, inplace=False))

def Deconv(n\_input, n\_output, k\_size=4, stride=2, padding=1):

return nn.Sequential(

nn.ConvTranspose2d(

n\_input, n\_output,

kernel\_size=k\_size,

stride=stride, padding=padding,

bias=False),

nn.BatchNorm2d(n\_output),

nn.ReLU(inplace=True))

class Generator(nn.Module):

def \_\_init\_\_(self, z=100, nc=64):

super(Generator, self).\_\_init\_\_()

self.net = nn.Sequential(

Deconv(z, nc\*8, 4,1,0),

Deconv(nc\*8, nc\*4, 4,2,1),

Deconv(nc\*4, nc\*2, 4,2,1),

Deconv(nc\*2, nc, 4,2,1),

nn.ConvTranspose2d(nc,3, 4,2,1,bias=False),

nn.Tanh()

)

def forward(self, input):

return self.net(input)

class Discriminator(nn.Module):

def \_\_init\_\_(self, nc=64):

super(Discriminator, self).\_\_init\_\_()

self.net = nn.Sequential(

nn.Conv2d(

3, nc,

kernel\_size=4,

stride=2,

padding=1, bias=False),

nn.LeakyReLU(0.2, inplace=True),

Conv(nc, nc\*2, 4,2,1),

Conv(nc\*2, nc\*4, 4,2,1),

Conv(nc\*4, nc\*8, 4,2,1),

nn.Conv2d(nc\*8, 1,4,1,0, bias=False),

nn.Sigmoid())

def forward(self, input):

return self.net(input)

device = torch.device('cuda')

dis\_model = Discriminator()

gen\_model = Generator()

gen\_model.apply(weights\_init)

dis\_model.apply(weights\_init)

dis\_model.to(device)

gen\_model.to(device)

print('init model')

Xác định tham số

from torch import optim

real\_label = 1.

fake\_label = 0.

lr = 0.0002

beta1 = 0.5

criterion = nn.BCELoss()

optim\_D = optim.Adam(dis\_model.parameters(), lr=lr, betas=(beta1, 0.999))

optim\_G = optim.Adam(gen\_model.parameters(), lr=lr, betas=(beta1, 0.999))

Train model

img\_list = []

G\_losses = []

D\_losses = []

iters = 1

epoch\_nb = 30

fixed\_noise = torch.randn(32, 100, 1,1, device=device)

from torch.distributions.uniform import Uniform

for epoch in range(epoch\_nb):

for i, data in enumerate(dataloader):

# Train Discriminator

## Train with real image

dis\_model.zero\_grad()

real\_img = data[0].to(device)

bz = real\_img.size(0)

# label smoothing

label = Uniform(0.9, 1.0).sample((bz,)).to(device)

output = dis\_model(real\_img).view(-1)

error\_real = criterion(output, label)

error\_real.backward()

D\_x = output.mean().item()

## Train with fake image

noise = torch.randn(bz, 100, 1,1, device=device)

fake\_img = gen\_model(noise)

label = Uniform(0., 0.05).sample((bz,)).to(device)

output = dis\_model(fake\_img.detach()).view(-1)

error\_fake = criterion(output, label)

error\_fake.backward()

D\_G\_z1 = output.mean().item()

error\_D = error\_real + error\_fake

optim\_D.step()

## Train Generator

gen\_model.zero\_grad()

label = Uniform(0.95, 1.0).sample((bz,)).to(device)

output = dis\_model(fake\_img).view(-1)

error\_G = criterion(output, label)

error\_G.backward()

optim\_G.step()

D\_G\_z2 = output.mean().item()

if i % 300 == 0:

print('[%d/%d][%d/%d]\tLoss\_D: %.4f\tLoss\_G: %.4f\tD(x): %.4f\tD(G(z)): %.4f / %.4f'

% (epoch, epoch\_nb, i, len(dataloader),

error\_D.item(), error\_G.item(), D\_x, D\_G\_z1, D\_G\_z2))

if epoch > 1:

if (iters % 1000 == 0) or ((epoch == epoch\_nb-1) and (i == len(dataloader)-1)):

with torch.no\_grad():

fake\_img = gen\_model(fixed\_noise).detach().cpu()

fake\_img = make\_grid(fake\_img, padding=2, normalize=True)

img\_list.append(fake\_img)

plt.figure(figsize=(10,10))

plt.imshow(img\_list[-1].permute(1,2,0))

plt.show()

iters += 1

Kết quả

A picture containing dark

Description automatically generated

# Chương 4. KẾT LUẬN

Như vậy các bạn đã tìm hiểu xong lý thuyết của model GAN. Mình xin tổng kết lại một số điểm chính:

* Ý tưởng của GAN dựa trên bài toán zero-sum game của lý thuyết trò chơi. Hai người chơi có lợi ích xung đột là Generator và Discriminator.
* Generator có mục tiêu là sinh ra bức ảnh giống với ảnh thật nhất. Đầu vào của Generator (trong kiến trúc GAN đầu tiên năm 2014) là một nhiễu được khởi tạo ngẫu nhiên.
* Discriminator có mục tiêu là phân biệt ảnh giả sinh ra từ Generator với ảnh thật.
* Hàm loss function lồng ghép đồng thời loss function của Generator và Discriminator. Loss function về bản chất vẫn là một hàm cross entropy của bài toán phân loại nhị phân.
* Quá trình huấn luyện GAN sẽ xen kẽ giữa Discriminator và Generator. Chúng ta sẽ huấn luyện trước Discriminator với k steps và cố định hệ số mô hình của Generator. Sau đó chúng ta huấn luyện Generator và cố định Discriminator.
* Ngoài ra GAN còn rất nhiều các biến thể khác như Wasserstein GAN, StarGAN, StyleGAN, SRGAN, CycleGAN, Pix2pix, BigGAN,…. rất rất nhiều các mô hình GAN khác chưa được liệt kê hết và vẫn đang tiếp tục được phát triển. Mình sẽ hướng dẫn các bạn ở những bài sau.

DANH SÁCH TÀI LIỆU THAM KHẢO

[1] https://nttuan8.com/bai-1-gioi-thieu-ve-gan/

[2] Data-Mining-and-Analysis\_-Fundamental-Concepts-and-Algorithms-Zaki-Meira-2014-05-12

[3] Christian W. Günther and Eric Verbeek (2012). XES: Standard Definition (Version 1.3). Technical Report, Eindhoven University of Technology

[4] https://towardsdatascience.com/image-generation-in-10-minutes-with-generative-adversarial-networks-c2afc56bfa3b