TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**CUỐI KỲ**

**HỌC MÁY**

*Người hướng dẫn*: **PGS.TS. LÊ ANH CƯỜNG**

*Người thực hiện*: **HOÀNG ĐÌNH QUÝ VŨ – 521H0517**

**LÊ HUỲNH HUYỀN TRANG - 520C0156**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

A red and blue logo

Description automatically generated

**CUỐI KỲ**

**HỌC MÁY**

*Người hướng dẫn*: **PGS.TS. LÊ ANH CƯỜNG**

*Người thực hiện*: **HOÀNG ĐÌNH QUÝ VŨ – 521H0517**

**LÊ HUỲNH HUYỀN TRANG - 520C0156**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

LỜI CẢM ƠN

Đây là phần tác giả **tự viết** ngắn gọn, thể hiện sự biết ơn của mình đối với những người đã giúp mình hoàn thành Luận văn/Luận án. Tuyệt đối không sao chép theo mẫu những “lời cảm ơn” đã có.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của **PGS.TS. Lê Anh Cường**. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Hoàng Đình Quý Vũ*

*Lê Huỳnh Huyền Trang*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

MỤC LỤC

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC KÝ HIỆU**

*f Tần số của dòng điện và điện áp (Hz)*

*p Mật độ điện tích khối (C/m3)*

**CÁC CHỮ VIẾT TẮT**

CSTD Công suất tác dụng

MF Máy phát điện

BER Tỷ lệ bít lỗi

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 2.1: Kiến trúc FTP 1](#_Toc387689394)

**DANH MỤC BẢNG**

[Bảng 3.1 Ví dụ cho chèn bảng 1](#_Toc387689363)

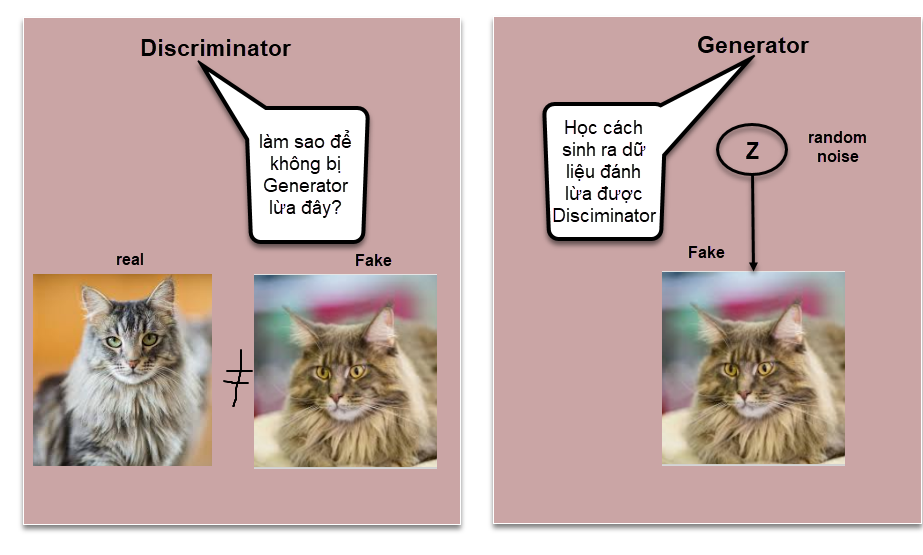
CHƯƠNG 1 – Kiến trúc và huấn luyện BERT

CHƯƠNG 2 – GAN

* 1. Giới thiệu

Mạng đối kháng sinh thành (GAN - Generative Adversarial Network) là một mô hình học máy sâu được sử dụng để tạo ra các dữ liệu mới, giống như thật. GAN bao gồm hai mạng thần kinh đối kháng nhau:

* Mạng sinh (Generator): Nhiệm vụ của mạng này là tạo ra các dữ liệu giả, ví dụ như hình ảnh, âm thanh, văn bản, từ một phân phối ngẫu nhiên.
* Mạng phân biệt (Discriminator): Nhiệm vụ của mạng này là phân biệt giữa dữ liệu thật và dữ liệu giả do mạng sinh tạo ra.



Ý tưởng của mô hình Generative Adversarial Network (GAN) bắt nguồn từ trò chơi zero-sum non-cooperative game, tương tự như cờ vua hoặc cờ tướng, trong đó một người chơi thắng sẽ khiến người kia thua. Cả Generator và Discriminator trong mạng GAN đều hoạt động như hai đối thủ trong trò chơi này. Trong lý thuyết trò chơi, mô hình GAN sẽ hội tụ khi cả Generator và Discriminator đạt tới trạng thái cân bằng Nash, tức là cả hai đối thủ đều không thể cải thiện tình hình của mình mà chỉ có thể cản trở đối thủ.

* 1. GANs sinh ra để giải quyết vấn đề gì?

Mạng Tạo Sinh Đối Kháng (GANs) được phát triển để giải quyết một số vấn đề lớn mà các mạng neural network truyền thống gặp phải:

* **Dễ bị lừa bởi nhiễu (noise)**: Các mạng neural network thông thường có thể dễ dàng bị đánh lừa khi thêm một lượng nhiễu nhỏ vào dữ liệu. Ví dụ, một hình ảnh mèo có thể được nhận dạng sai là một con chó nếu thêm một lượng nhiễu không đáng kể.
* **Dễ bị overfitting**: Khi huấn luyện trên một lượng dữ liệu hạn chế, các mạng neural network thông thường có xu hướng học quá mức các đặc trưng của dữ liệu huấn luyện, làm giảm khả năng tổng quát hóa cho dữ liệu mới.
* **Ánh xạ tuyến tính giữa Input và Output**: Trong nhiều trường hợp, quá trình ánh xạ từ đầu vào (Input) sang đầu ra (Output) của các mạng neural network là tuyến tính. Điều này không phản ánh đúng sự phức tạp của dữ liệu thực tế, nơi mà các đặc trưng có thể thay đổi phi tuyến tính và phức tạp.

2.3 Nguyên lý hoạt động của GAN

Như ở trên mình cũng đã giới thiệu về Generator và Discriminator. Thì 2 khái niệm:

* Mạng sinh (Generator): Nhiệm vụ của mạng này là tạo ra các dữ liệu giả, ví dụ như hình ảnh, âm thanh, văn bản, từ một phân phối ngẫu nhiên. 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.
* Mạng phân biệt (Discriminator): Nhiệm vụ của mạng này là phân biệt giữa dữ liệu thật và dữ liệu giả do mạng sinh tạo ra. Vì vậy, Discriminator cố gắng phân biệt dữ liệu mà Genarator đổ vào và chứng minh nó là hàng fake sau đó thông báo lại cho Generator để nó cải thiện, cứ như thế quá trình này lặp đi lặp lại để generator có thể tạo ra sample hoàn hảo nhất mà Discriminator không thể phân biệt được.

A diagram of a network

Description automatically generated

2.3.1 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. Từ tập véc tơ đầu vào 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.

A diagram of a cat

Description automatically generated

* + 1. Discriminator

Mô hình Discriminator sẽ có tác dụng phân biệt ảnh input là thật hay giả. Nhãn của mô hình sẽ là *thật* nếu 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.

A diagram of a cat and dog

Description automatically generated

* + 1. Loss function

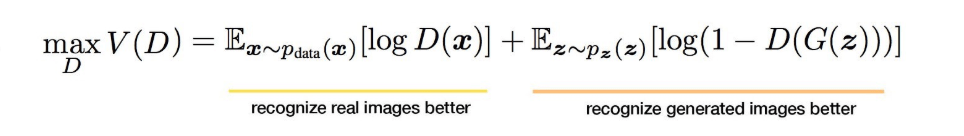
Kí hiệu:

* z là noise đầu vào của generator, x là dữ liệu thật từ bộ dataset.
* G là mạng Generator, D là mạng Discriminator.
* G(z) là ảnh được sinh ta từ Generator.
* D(x) là giá trị dự đoán của Discriminator xem ảnh x là thật hay không
* D(G(z)) là giá trị dự đoán xem ảnh sinh ra từ Generator là ảnh thật hay không.

Vì ta có 2 mạng Generator và Discriminator với mục tiêu khác nhau, nên cần thiết kế 2 loss function cho mỗi mạng.

**Discriminator** thì cố gắng phân biệt đâu là ảnh thật và đâu là ảnh giả. Vì là bài toán binary classification nên loss function dùng giống với **binary cross-entropy loss** của bài sigmoid.

Giá trị output của model qua hàm sigmoid nên sẽ trong (0, 1) nên Discriminator sẽ được train để input ảnh ở dataset thì output gần 1, còn input là ảnh sinh ra từ Generator thì output gần 0(Giả), hay D(x) -> 1 (Thật) còn D(G(z)) -> 0. Hay nói cách khác là loss function muốn maximize D(x) và minimize D(G(z)) = maximize (1 – D(G(z))). E là kì vọng, hiểu đơn giản là lấy trung bình của tất cả dữ liệu. Do đó loss function của Discriminator trong paper gốc được viết lại thành.



Generator sẽ học để đánh lừa Discriminator rằng số nó sinh ra là số thật, hay D(G(z)) -> 1. Hay loss function muốn maximize D(G(z)), tương đương với minimize (1 – D(G(z))).

A black text on a white background

Description automatically generated

Gộp lại ta có loss của mô hình GAN:

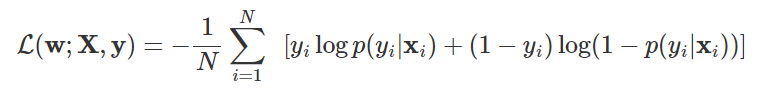
A close-up of a word

Description automatically generated

Từ hàm loss của GAN có thể thấy là việc train Generator và Discriminator đối nghịch nhau, trong khi D cố gắng maximize loss thì G cố gắng minimize loss. Quá trình train GAN kết thúc khi model GAN đạt đến trạng thái cân bằng của 2 models, gọi là Nash equilibrium.

**Giải thích thêm:**

* Vì tại sao có công thức của discriminator. Như ở trên đã nói thì loss của discrimator giống với **binary cross-entropy loss.** Công thức của binary cross-entropy:



* là xác suất dự báo nhãn từ mô hình. Vì vậy D(x) cũng tương tự nó cũng là dự báo xác suất cho đầu vào x (). Vì vậy nó có 2 trường hợp xãy ra là = 1 va = 0 (tương đương với Thật hay Giả).
  + Với và 1- = 0 thay vào công thức cross-entropy thì nó còn lại
  + Tương tự với và 1- = 1,
  1. Ứng dụng

Về các ứng dụng của GAN các bạn có thể kể đến những lĩnh vực chính sau:

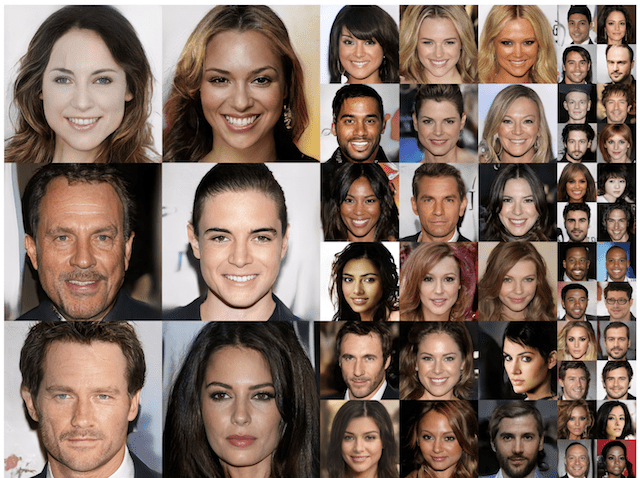
1. Tạo hình ảnh mới: GANs có khả năng sinh ra các hình ảnh chất lượng cao từ nhiễu ngẫu nhiên. Chúng có thể được sử dụng để tạo ra chân dung người, phong cảnh, hoặc các hình ảnh nghệ thuật.
2. Chuyển đổi phong cách hình ảnh: CycleGAN có thể chuyển đổi phong cách của một hình ảnh từ một miền này sang miền khác, như chuyển từ ảnh mùa hè sang mùa đông.
3. Tạo dữ liệu tổng hợp: Trong các lĩnh vực như y học hoặc an ninh, GANs có thể được sử dụng để tạo ra dữ liệu tổng hợp để huấn luyện các mô hình học máy khi dữ liệu thật khó thu thập.
4. Nâng cấp hình ảnh (Super-Resolution): GANs có thể nâng cấp độ phân giải của hình ảnh bằng cách thêm chi tiết và làm sắc nét hình ảnh.
5. Tạo video giả lập: GANs có thể được sử dụng để tạo ra các video giả lập từ dữ liệu đầu vào nhất định, ứng dụng trong giải trí và truyền thông.
6. Tạo âm thanh và nhạc: GANs cũng có thể được áp dụng để tạo ra các mẫu âm thanh hoặc nhạc mới.

Và dưới đây là link tham khảo ở rất nhiều trang khác nhau như **18 ứng dụng** của GAN - machine learning mastery, Các bạn có thể tìm đọc, trong là các mục và paper viết về các ứng dụng đó.

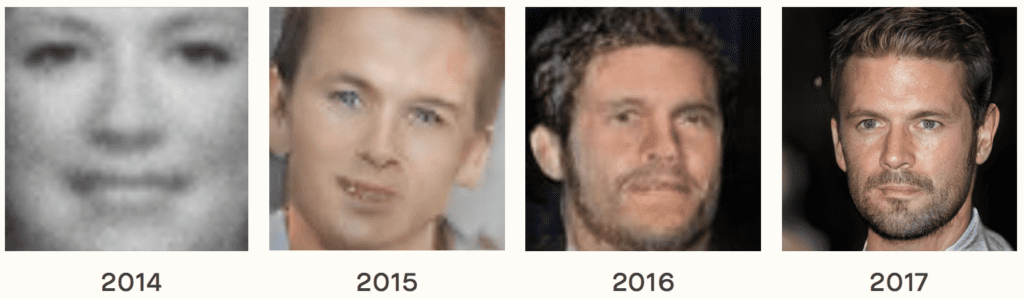
[**https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/**](https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/)

Điển hình như:

* Tạo ra khuôn mặt người: GAN có khả năng tạo ra những khuôn mặt nhân tạo mà rất khó phân biệt với người thật. Chất lượng của những model GAN áp dụng trên khuôn mặt ngày càng tốt hơn qua từng năm.



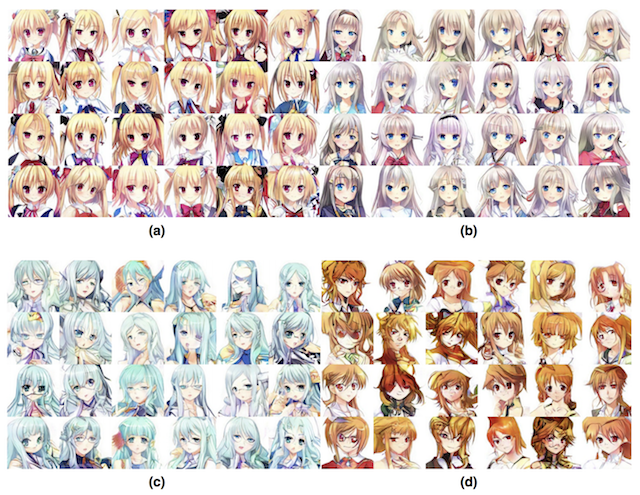
* Thay đổi độ tuổi của khuôn mặt: Chắc hẳn các bạn đã không còn xa lạ với ứng dụng thay đổi tuổi của khuôn mặt. Dựa trên khuôn mặt của bạn hiện tại, GAN sẽ sinh ra các biến thể theo từng độ tuổi của bạn.



* Sinh ảnh các vật thể Tất nhiên những gì mà GAN đã thực hiện trên con người thì nó có thể ứng dụng được trên những loài động vật khác.



* Tạo nhân vật hoạt hình Có lẽ ngành công nghiệp phim hoạt hình Nhật Bản



* 1. Chú ý

Bên cạnh những ứng dụng ở trên 1 vấn đề cũng đáng quan ngại trong lĩnh vực này đó là Deep fake. Cụm từ "deep fake" đã trở nên rất phổ biến trong vài năm gần đây. Hãy tưởng tượng bạn có thể đóng vai chính trong bộ phim yêu thích của mình - điều này không còn là giấc mơ xa vời nữa nhờ vào công nghệ deep fake. Tuy nhiên, khi nhắc đến công nghệ này, vấn đề đầu tiên chúng ta phải quan tâm không phải là công nghệ mà là các khía cạnh đạo đức. Công nghệ này nên được sử dụng cho mục đích nghiên cứu và sáng tạo, chứ không phải cho các mục đích xấu như khiêu dâm hoặc thao túng chính trị.

Dưới đây là ví dụ:

A person in a suit and tie

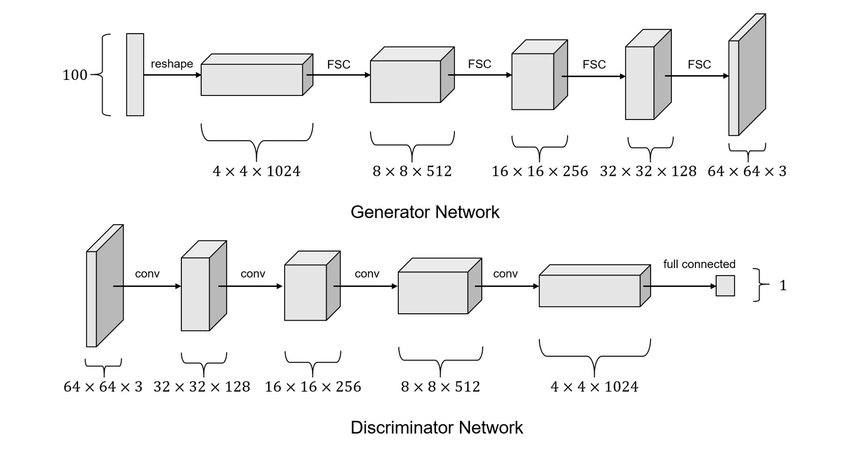
Description automatically generated

A diagram of a person's face

Description automatically generated

* 1. DCGAN

Deep Convolutional GAN (DCGAN) là một biến thể của Mạng Tạo Sinh Đối Kháng (GAN), được giới thiệu bởi Alec Radford, Luke Metz, và Soumith Chintala vào năm 2015. DCGAN sử dụng các mạng neuron tích chập (CNNs) trong cả Generator và Discriminator để cải thiện chất lượng hình ảnh sinh ra. Như đã biết rằng, CNN xử lý dữ liệu ảnh tốt hơn và hiệu quả hơn rất nhiều so với Neural Network truyền thống. Vì vậy thay vì sử mô hình Generator và Discriminator sử dụng neural network (các fully connected layer) như trong GAN thì ở DCGAN, họ xây dự bằng mô hình CNN với 2 layers chính là **convolutional layer** và **transposed convolutional layer**.



Điểm khác biệt mà website “paperwithcode” có nói:

* Thay thế bất kỳ lớp pooling nào bằng các tích chập strided (discriminator) và tích chập fractional-strided (generator).
* Sử dụng batchnorm cả trong generator và discriminator. Loại bỏ các lớp ẩn kết nối đầy đủ cho các kiến trúc sâu hơn.
* Sử dụng ReLU activation trong generator cho tất cả các lớp trừ đầu ra, sử dụng tanh.
* Sử dụng LeakyReLU activation trong discriminator cho tất cả các lớp.
  + 1. Generator

Áp dụng mạng DCGAN với dữ liệu CIFAR-10. CIFAR-10 dataset bao gồm 60000 ảnh màu kích thước 32×32 thuộc 10 thể loại khác nhau. Mỗi thể loại có 6000 ảnh.

A collage of different animals

Description automatically generated

Mạng Generator nhằm mục đích sinh ảnh fake, input là noise vector kích thước 128 và output và ảnh fake cùng kích thước ảnh thật (32 \* 32 \*3).

A diagram of a generator

Description automatically generated

Các layer trong mạng:

* Dense (fully connected) layer: 128\*1 -> 2048\*1
* Flatten chuyển từ vector về dạng tensor 3d, 2048\*1 -> 2\*2\*512
* Transposed convolution stride=2, kernel=256, 2\*2\*512 -> 4\*4\*256
* Transposed convolution stride=2, kernel=128, 4\*4\*256 -> 8\*8\*128
* Transposed convolution stride=2, kernel=64, 8\*8\*128 -> 16\*16\*64
* Transposed convolution stride=2, kernel=3, 16\*16\*64 -> 32\*32\*3

Chú ý khi chiều cao và chiều rộng tăng, thì độ sâu sẽ giảm và ngược lại.

2.6.1.1 Transposed convolution

Ở đây, em sẽ giải thích thêm về transposed convolution. Transposed convolution hay deconvolution có thể coi là phép toán ngược của convolution.

* Nếu như **convolution** với stride > 1 giúp làm **giảm** kích thước của ảnh thì **transposed convolution** với stride > 1 sẽ làm **tăng** kích thước ảnh.
* Ví dụ stride = 2 và padding = ‘SAME’ sẽ giúp **gấp đôi** width, height kích thước của ảnh.

Transposed convolution có 2 kiểu định nghĩa. Với Kiểu 1:

Ví dụ với Convolution với stride (s) = 1 và padding (p) = 0 và kernel.

A whiteboard with writing on it

Description automatically generated

Ví dụ về Transposed convolution với s = 1, p = 0.

**A diagram of a number

Description automatically generated with medium confidence**

Hay nếu ta thay đổi với s = 1, p = 1. Thì nó sẽ loại bỏ cạnh ngoài của output và nó chỉ còn lại là ma trận 1 x 1 = [4]. Hay dưới đây là ví dụ khác với s = 2, p = 1:

A drawing of a grid with red squares and green squares

Description automatically generated

Với kiểu 2 ta có ví dụ dưới đây:

A diagram of a transposed convolution

Description automatically generated

Và với 2 công thức cần biết

* z = s - 1: giá trị cho biết chèn giữa hàng vào cột
* p’ = k – p – 1: padding thêm vào input (convolution).

Với ví dụ là s = 1, p = 0 (stride và padding), k = 3 thì khi ta convolution (lớn sang nhỏ ma trận). kernel = 3 x 3 -> k = 3

* z = s – 1 = 1 – 1 = 0: chèn vào hàng/ cột ở giữa (giá trị 0 ý là không có chèn vào).
* p’ = k – p – 1 = 3 – 0 – 1 = 2: padding input là 2.
* Màu xanh dương là **input**, xanh lá cây đậm là **output**. Nếu check lại thì phép tính convolution với input 5\*5, kernel size 3\*3, s = 2, p = 1 sẽ được output kích thước 3\*3.

A blue and green squares

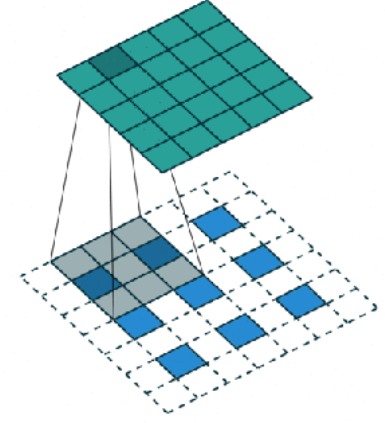
Description automatically generated A blue squares with a blue grid

Description automatically generated with medium confidence

Với ví dụ khác là s = 2, p = 1 (stride và padding), k = 3.

* z = s – 1 = 2 – 1 = 1: chèn vào hàng/ cột ở giữa.
* p’ = k – p – 1 = 3 – 1 – 1 = 1: padding input là 1.
* Màu xanh dương là **input**, xanh lá cây đậm là **output**. Tính convolution với input 5\*5, kernel size 3\*3, s = 2, p = 1 sẽ được output kích thước 3\*3. Ngược lại là transposed convolution.

A blue and green squares

Description automatically generated 

https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

* + 1. Discriminator

Mạng Discriminator nhằm mục đích phân biệt ảnh thật từ dataset và ảnh fake do Generator sinh ra, input là ảnh kích thước (32 \* 32 \*3), output là ảnh thật hay fake (binary classification).

A diagram of a diagram of a diagram

Description automatically generated

Mô hình discriminator đối xứng lại với mô hình generator. Ảnh input được đi qua convolution với stride = 2 để giảm kích thước ảnh từ 32\*32 -> 16\*16 -> 8\*8 -> 4\*4 -> 2\*2. Khi giảm kích thước thì depth tăng dần. Cuối cùng thì tensor shape 2\*2\*512 được reshape về vector 2048 và dùng 1 lớp fully connected chuyển từ 2048d về 1d.

Loss function giống với GAN nên nhóm em sẽ không nhắc lại.

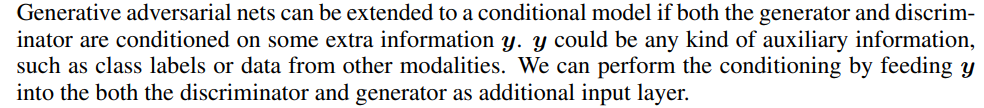
* 1. cGAN

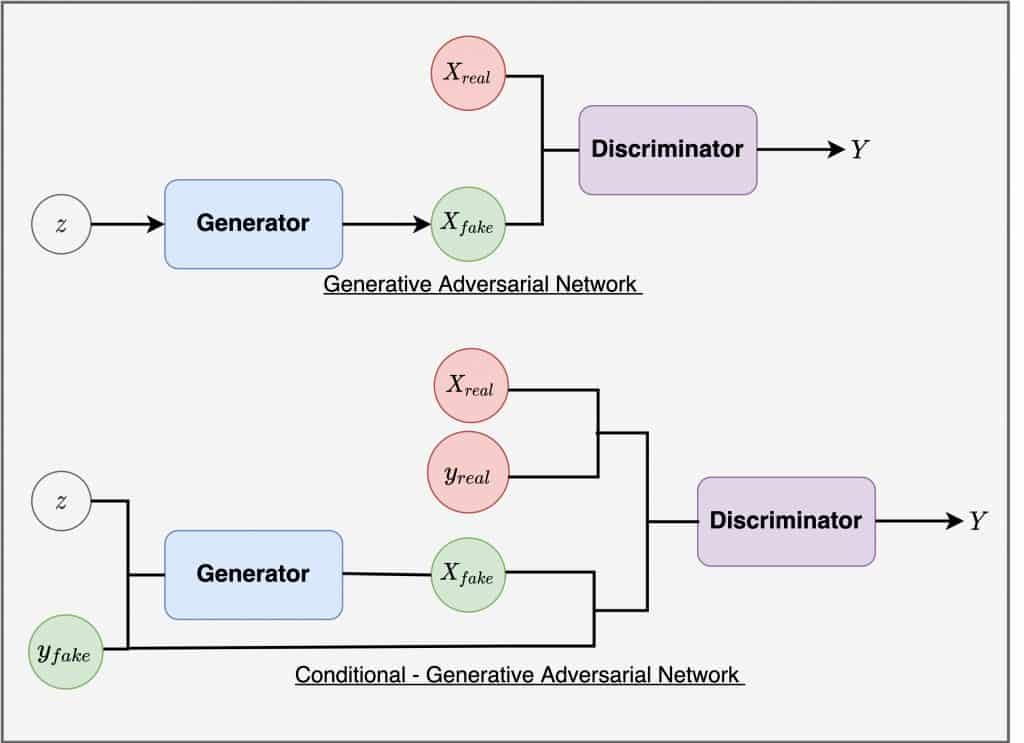
DCGAN (GAN tích chập sâu) là một GAN mô hình được sử dụng trong các hình ảnh xử lý tác vụ. Bài viết này được viết cho thị giác máy tính, vì vậy chúng tôi sẽ sử dụng thuật ngữ DCGAN thay vì GAN.

Nhược điểm của DCGAN là chúng tôi không thể kiểm soát hình ảnh được tạo ra thuộc về bất kỳ lớp nào, vì nó được tạo hoàn toàn ngẫu nhiên. Ví dụ, trong bài viết trước, khi bạn truyền vào mạng một véc tơ noise 𝑧 ngẫu nhiên, mỗi lần suy luận có thể tạo ra một số chữ số khác nhau. Điều này khiến chúng ta khó biết trước hình ảnh cần thuộc về lớp nào và đây cũng là một chế độ hạn chế của DCGAN.

cGAN sẽ giúp họ tạo ra các hình ảnh thuộc về một lớp cụ thể công cụ mà họ có thể có ý muốn dựa trên thông tin bổ sung được thêm vào mô hình, đó là nhãn y. Đây được coi là một điều kiện để tạo ra hình ảnh, vì vậy mô hình được gọi là GAN ​​có điều kiện (conditional GAN - cGAN).

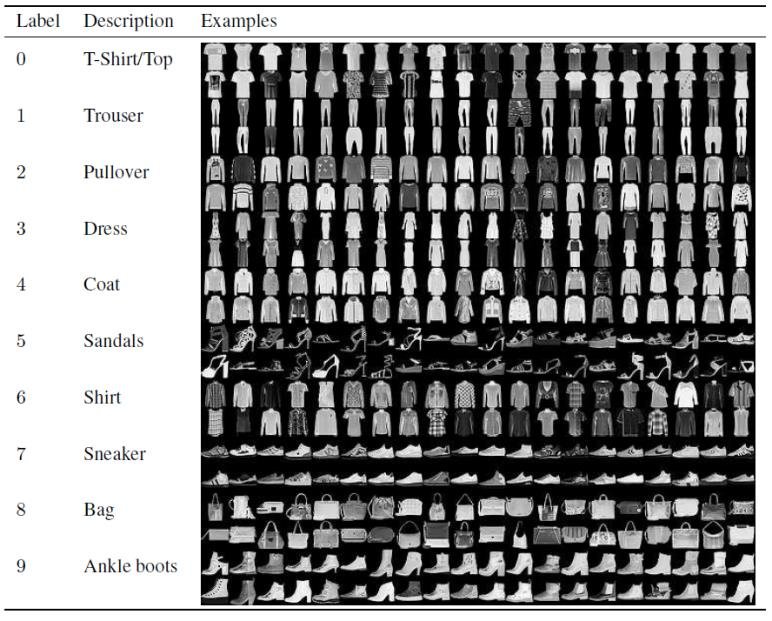
Trích dẫn từ paper “[Conditional Generative Adversarial Nets, 2014](https://arxiv.org/abs/1411.1784)”





2.7.1 Generator

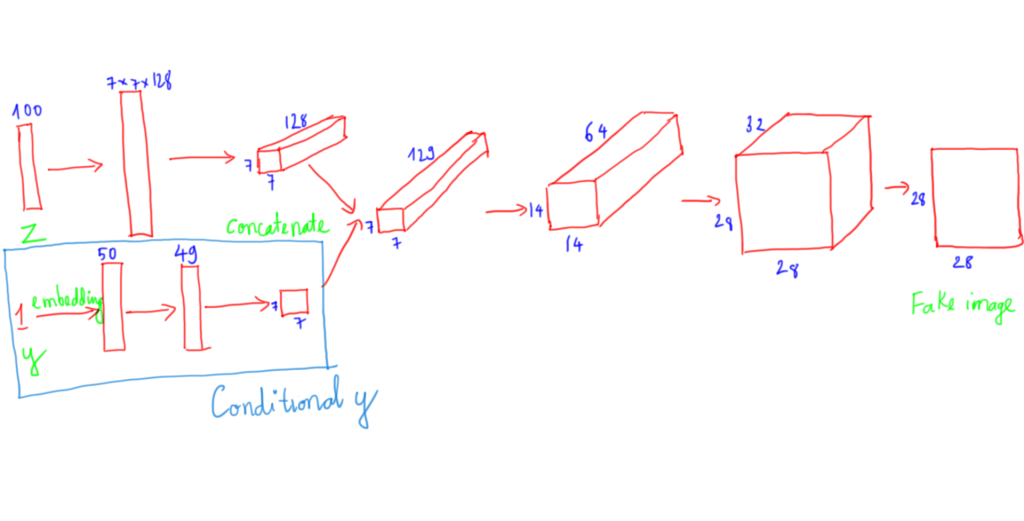
Trước khi đi vô generator thì nhóm em sẽ nói sơ lại về dư liệu fashion-mnist. Dữ liệu Fashion-MNIST về quần áo, giày dép gồm 60000 ảnh training và 10000 ảnh test. Ảnh xám kính thước 28\*28 thuộc 10 lớp khác nhau.



Như phần trên, Generator đã tạo ra ảnh giả từ vector noise. Một điểm mạnh của vector ngẫu nhiên là chúng ta có thể lấy mẫu được nhiều giá trị khác nhau, điều này có nghĩa là sau khi huấn luyện mạng GAN, chúng ta có thể tạo ra nhiều hình ảnh giả khác nhau. Tuy nhiên, chúng ta không thể kiểm soát được loại hình ảnh được tạo ra (áo, giày, dép,…).

Ngoài vector noise z, chúng ta sẽ thêm vào y, là một số từ 0 đến 9, mỗi loại trong dữ liệu Fashion-MNIST sẽ được mã hóa thành một số. Mục tiêu của chúng ta là vector z kết hợp với số y sẽ tạo ra dữ liệu tương ứng với loại đó.

Ứng với bài toán cho dư liệu fashion-mnist ta có kiến trúc của generator như sau:



* Đầu vào z là một vector noise như trước, sau đó được đưa qua một lớp dense để chuyển về kích thước 7\*7\*128, sau đó reshape thành tensor 3D kích thước 7x7x128 (**y1**).
* Đầu vào y là một số được đi qua lớp embedding của keras, lớp này tương tự như một bảng ánh xạ mỗi số thành một vector 50\*1, sau đó được đưa qua lớp dense với output 49 node cuối cùng được reshape thành tensor 3D kích thước 7x7x1 (**y2**).
* Sau đó, y1 và y2 được xếp chồng lên nhau để tạo thành tensor 3D kích thước 7\*7\*129, tiếp theo đi qua lớp transposed convolution để tăng kích thước lên 14\*14 và 28\*28, cuối cùng cho ra output 28\*28\*1.

2.7.2 Discriminator

Tương tự như ở trong generator bên cạnh ảnh, nó sẽ thêm vào y, 1 số từ (0 – 9) tương ứng với thể loại trong dữ liệu Fashion-MNIST.

A diagram of a diagram

Description automatically generated

* Đầu vào y là một số được đưa qua lớp nhúng của keras, lớp này tương tự như một bảng ánh xạ mỗi số thành một vector có kích thước 50\*1.
* Sau đó, vector này được đưa qua lớp dense với 28\*28 node đầu ra. Kết quả cuối cùng được chuyển đổi về dạng tensor 3 chiều có kích thước 28x28x1 (y1).
* Sau đó, y1 và ảnh đầu vào được xếp chồng lên nhau để tạo ra một tensor 3 chiều có kích thước 28\*28\*2.
* Tensor này sau đó được đưa qua lớp convolution với bước nhảy (stride) bằng 2 để giảm kích thước của ảnh từ 28\*28 xuống còn 14\*14 và sau đó là 7\*7. Khi giảm kích thước, độ sâu của tensor tăng dần.
* Cuối cùng, tensor có kích thước 2\*2\*512 được chuyển đổi về dạng vector có kích thước 2048 và đi qua một lớp fully connected để chuyển đổi từ 2048 chiều về 1 chiều.

2.7.3 Loss function

Model cGAN cũng có loss function tương tự như model DCGAN (giống gan) bao gồm loss function của model discriminator và loss function của model generator.



A diagram of a diagram

Description automatically generated

TÀI LIỆU THAM KHẢO

**PHỤ LỤC**

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh. . . . nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn