TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO: THỊ GIÁC MÁY TÍNH
Đề tài: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

PHẠM HOÀNG

hoang.pham170761@sis.hust.edu.vn

Ngành Kỹ thuật điều khiển và tự động hoá

Chuyên ngành Điều khiển tự động

Giảng viên hướng dẫn:	TS. Phạm Văn Trường
Bộ môn:	Điều khiển tự động
Viện:	Điện

HÀ NỘI, 01/2021

Mục lục

Lời nói đầu **3**

Chương 1. Giới thiệu về mạng GAN **4**

1.1. Mạng GAN là gì? 4

1.2. Cấu trúc của mạng GAN 4

1.2.1. Generator 6

1.2.2. Discriminator 7

1.2.3. Hàm mất mát 9

Chương 2. Cấu trúc mạng Deep Convolutional GAN **10**

2.1. Generator 10

2.2. Discriminator 11

Chương 3. Lập trình và kết quả **11**

3.1. Lập trình bài toán 11

3.2. Kết quả thực nghiệm 14

Danh mục tài liệu tham khảo **16**

Lời nói đầu

Thị giác máy tính là một lĩnh vực bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh. Về lĩnh vực nghiên cứu của thị giác máy tính thì rất là rộng với nhiều bài toán khác nhau, có thể kể đến như: bài toán nhận dạng đối tượng, bài toán phân vùng ảnh, ... ứng dụng rất nhiều trong các lĩnh vực giáo dục, an ninh, y tế, …

Vậy làm thế nào để máy tính có thể tự động vẽ một bức chân dung, hoặc thậm chí làm cho máy tính có thể tự động sáng tác ra một bản nhạc theo một phong cách nào đó - điều mà có vẻ như chỉ con người mới có thể làm được? Đó chính là sử dụng mạng GAN (Generative Adversarial Networks). Trong dự án này, em sẽ thực hiện sinh dữ liệu bằng cách sử dụng mạng GAN dựa trên dữ liệu cho trước.

Em xin cảm ơn sự hướng dẫn, hỗ trợ tận tình của thầy - TS. Phạm Văn Trường đã giúp em hoàn thành dự án này. Trong thời lượng hạn chế của học phần, với kỹ năng và kinh nghiệm ít ỏi, em không thể tránh khỏi những sai sót, rất mong được sự góp ý của thầy để dự án được hoàn thiện hơn.

Sinh viên thực hiện
Phạm Hoàng

Chương 1. Giới thiệu về mạng GAN

1.1. Mạng GAN là gì?

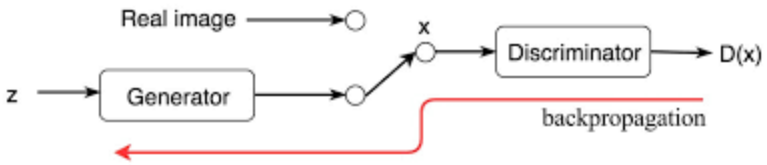
GAN (viết tắt của Generative Adversarial Networks) là một mạng nơ-ron nhân tạo có khả năng sinh ra dữ liệu (generative) mới.

Hình 1. Mặt người do mạng GAN sinh ra nhưng không phải là thật

1.2. Cấu trúc của mạng GAN

Mạng GAN được hình thành từ hai mạng nơ-ron khác nhau, bao gồm Generator và Discriminator. Hai mạng này có chức năng đối nghịch (adversarial) nhau. Trong khi Generator sinh ra các dữ liệu giống như thật (dữ liệu giả) thì Discriminator cố gắng phân biệt đâu là dữ liệu thật đã có và đâu là dữ liệu giả.

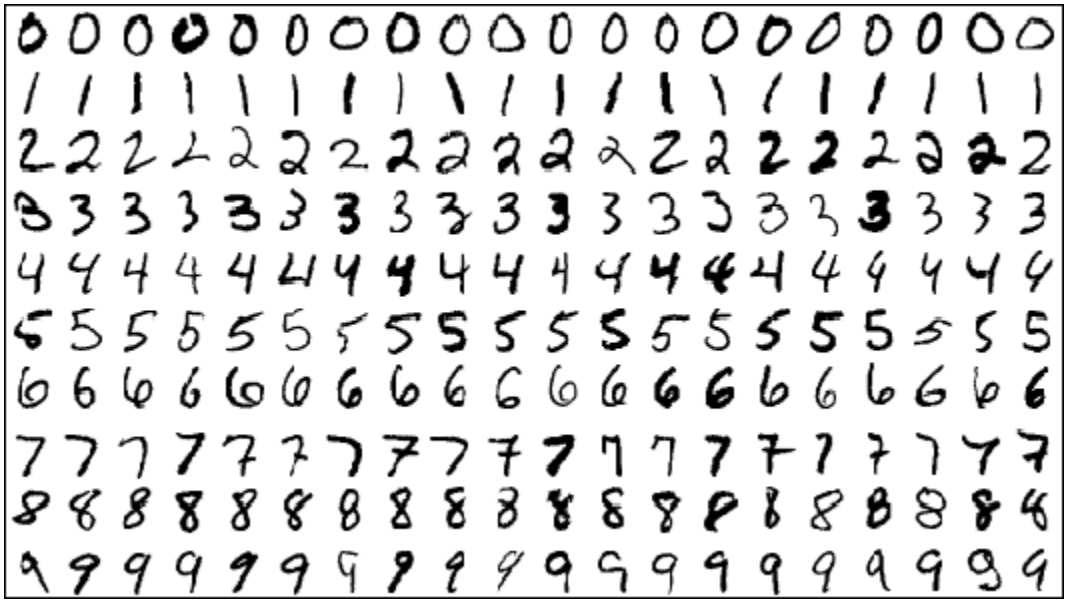
Ý tưởng của mạng GAN bắt nguồn từ thuyết non-cooperative game, còn gọi là zero-sum game, nghĩa là trò chơi có tổng bằng 0. Lấy ví dụ trong các trò chơi đối kháng như đánh cờ, nếu một người thắng thì người còn lại sẽ thua. Tuy nhiên cũng có một số trường hợp cả hai người bất phân thắng bại (hòa cờ), lúc đó trò chơi sẽ đạt đến trạng thái cân bằng (Nash equilibrium), không cạnh tranh (non-cooperative) nữa.



Hình 2. Nhiệm vụ của Generator và Discriminator

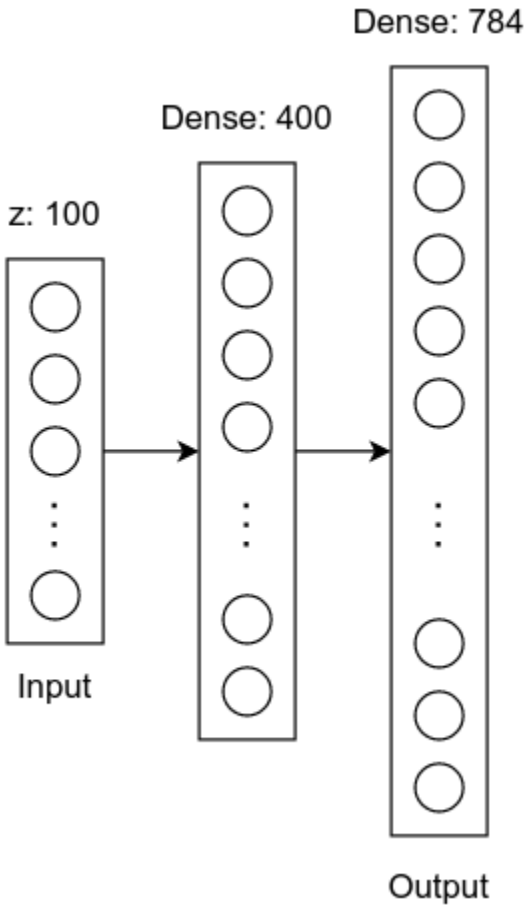
Generator và Discriminator trong mạng GAN được hiểu là hai đối thủ trong một trò chơi đối kháng. Một bên là Generator cố gắng sinh ra dữ liệu giả và đánh lừa Discriminator. Bên còn lại là Discriminator phải cố gắng phân biệt được dữ liệu mà Generator sinh ra là giả với dữ liệu thật đã có. Như vậy việc huấn luyện mạng sẽ hội tụ nếu như cả Generator và Discriminator đạt đến trạng thái cân bằng.

Bài toán sinh chữ số viết tay được trình bày dưới đây sẽ mô tả chi tiết hơn cấu trúc của Generator và Discriminator.



Hình 3. Bộ cơ sở dữ liệu các chữ số viết tay MNIST

1.2.1. Generator



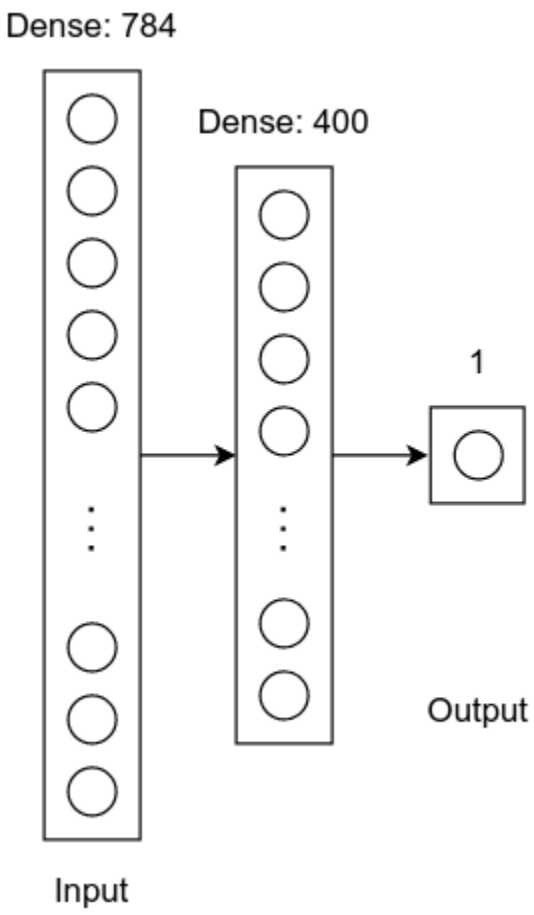
Hình 4. Cấu trúc của Generator

Mỗi ảnh trong cơ sở dữ liệu MNIST biểu diễn 1 số tự nhiên từ 0 - 9, có kích thước 28 x 28 = 784 (pixel). Nhiệm vụ của Generator là tạo ra dữ liệu giả (ảnh fake) giống với dữ liệu thật. Như vậy đầu ra của Generator có thể chọn là một lớp có 784 nơ-ron.

Generator có đầu vào là nhiễu (noise), tức là một vector bất kỳ, khi đó với mỗi noise vector khác nhau ta có thể sinh ra một chữ số, ở đây chọn noise vector có chiều bằng 100, tức lớp đầu vào có 100 nơ-ron (có thể tùy chỉnh).

Ngoài ra, ở đây Generator còn có thêm một lớp ẩn (hidden) ở giữa, kết nối fully-connected với đầu vào và đầu ra, có số lượng nơ-ron là 400.

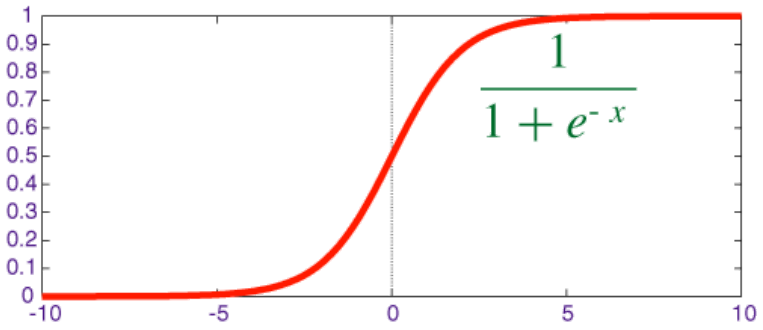
1.2.2. Discriminator



Hình 5. Cấu trúc của Discriminator

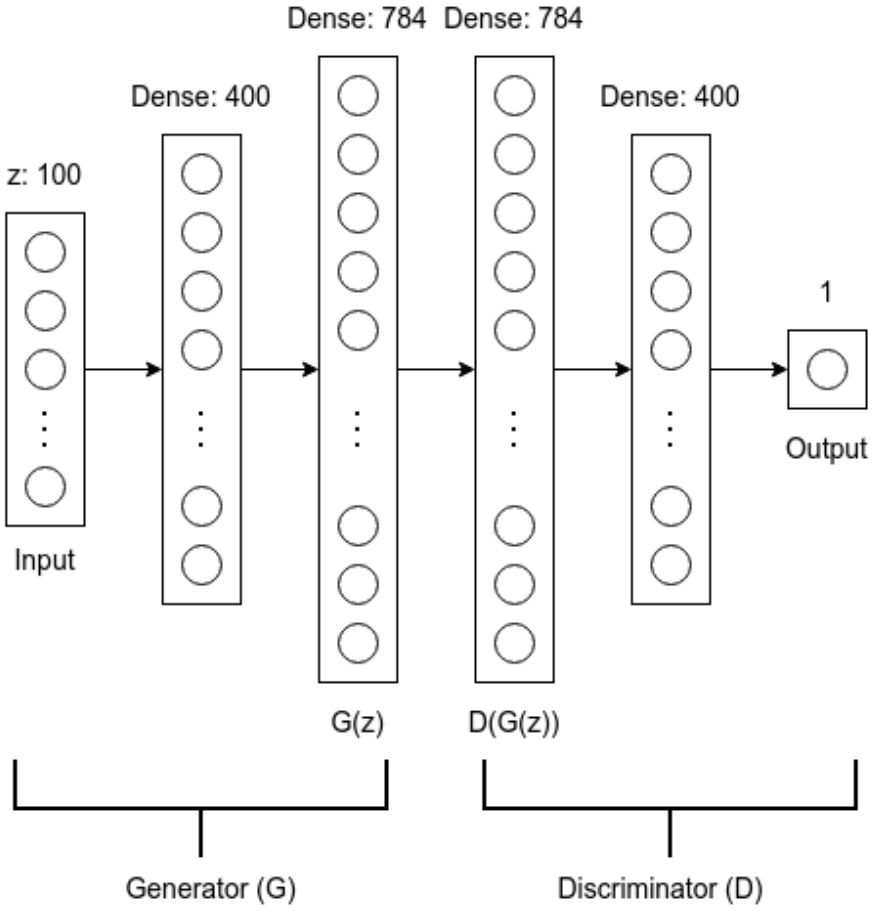
Nhiệm vụ của Discriminator là cố gắng phân biệt đâu là dữ liệu thật (ảnh gốc trong dataset) và đâu là dữ liệu giả (ảnh sinh từ Generator). Như vậy đầu vào của Discriminator là ảnh có kích thước 784, tương ứng với lớp đầu vào là 784 nơ-ron.

Đầu ra của Discriminator là phân bố xác suất, khi đó có thể sử dụng hàm kích hoạt sigmoid và chuyển về bài toán phân loại nhị phân. Trong đó: xác suất càng gần 1 tương ứng với ảnh thật và xác suất càng gần 0 tương ứng với ảnh giả.



Hình 6. Công thức và đồ thị của hàm sigmoid

Ngoài ra, ở đây Discriminator còn có thêm một lớp ẩn (hidden) ở giữa, kết nối fully-connected với đầu vào và đầu ra, có số lượng nơ-ron là 400. Như vậy có thể thấy, Generator và Discriminator có cấu trúc đối xứng nhau.



Hình 5. Cấu trúc của mạng GAN đơn giản

1.2.3. Hàm mất mát

Trước hết, ta quy ước một vài các ký hiệu để dễ dàng biểu diễn công thức toán học về sau:

Ký hiệu	Ý nghĩa
G	Generator
D	Discriminator
z	nhiễu để tạo ảnh giả
x	ảnh thật
G(z)	ảnh giả
D(x)	xác suất mà Discriminator dự đoán là ảnh thật
D(G(z))	xác suất mà Discriminator dự đoán là ảnh giả

Nhiệm vụ của Discriminator là phải phân biệt được đâu là ảnh thật và đâu là ảnh giả. Nghĩa là chúng ta cần phải huấn luyện Discriminator đủ tốt để nó có thể dự đoán $D(x) \rightarrow 1$ và $D(G(z)) \rightarrow 0$.

Khi đó để tối ưu hàm mất mát thì cần phải tối ưu cực đại *D(x)* và tối ưu cực tiểu *D(G(z))*. Việc này tương đương với việc tối ưu cực đại *D(x)* và tối ưu cực đại *(1 - D(G(z)))* (do xác suất nằm trong đoạn [0;1]).

Từ đó ta có hàm mất mát của mạng Discriminator được định nghĩa như sau:

$$\max_D V(D) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Với Generator, chúng ta cần phải huấn luyện để nó có thể đánh lừa được Discriminator, nghĩa là Discriminator chấp nhận ảnh giả là ảnh thật ($D(G(z)) \rightarrow 1$). Khi đó để tối ưu hàm mất mát thì cần phải tối ưu cực đại *D(G(z))*. Việc này tương đương với việc tối ưu cực tiểu *(1 - D(G(z)))*.

Từ đó ta có hàm mất mát của mạng Generator được định nghĩa như sau:

$$\min_G V(G) = \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

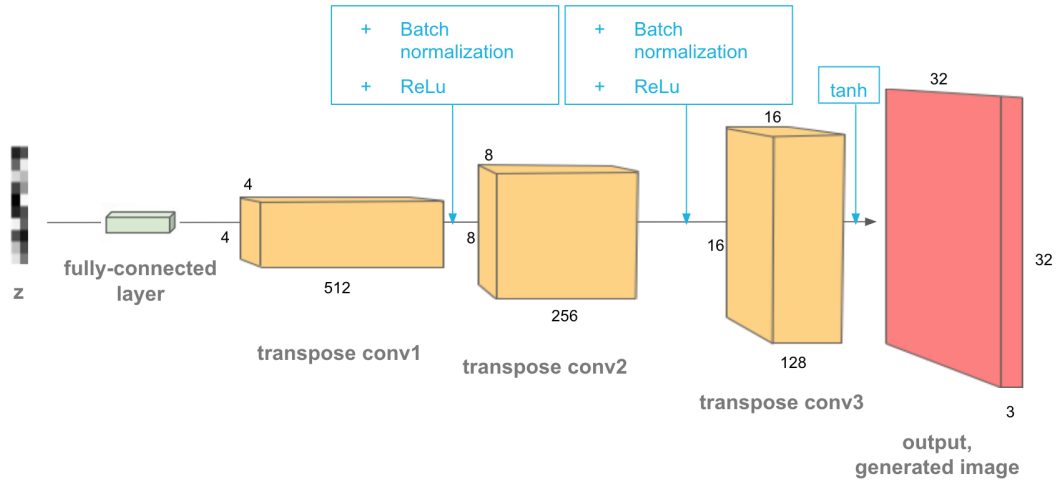
Tổng hợp lại ta có hàm mất mát của mạng GAN:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Chương 2. Cấu trúc mạng Deep Convolutional GAN

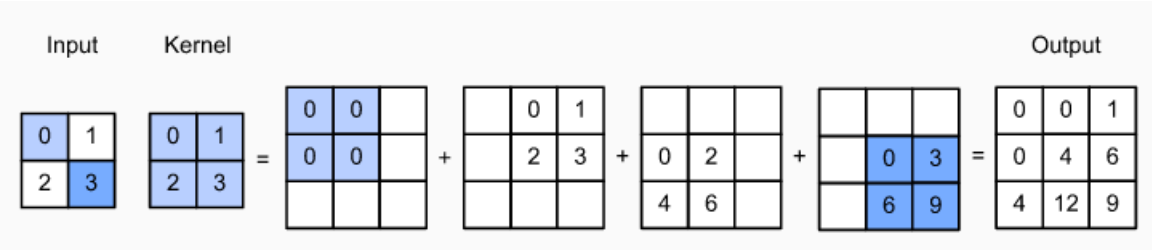
Ở chương 1, chúng ta đã cùng tìm hiểu về GAN với cấu trúc mạng MLP (Multi-layer Perceptron) là cấu trúc mạng nơ-ron truyền thống. Tuy nhiên bất lợi của mạng nơ-ron truyền thống là số lượng tính toán nhiều do kết nối fully-connected. Chính vì thế ở chương 2 này đề xuất phương án sử dụng GAN với cấu trúc mạng tích chập (CNN) để giảm trọng số, giảm khối lượng tính toán, ... còn gọi là mạng DCGAN (Deep Convolutional GAN).

2.1. Generator



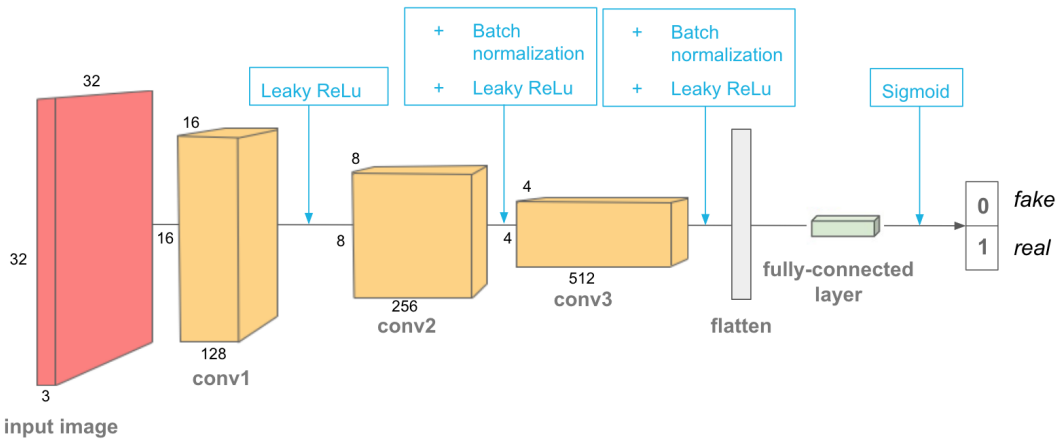
Hình 6. Cấu trúc của mạng DCGAN Generator

Trong cấu trúc Generator của DCGAN, ta sử dụng phép tích chập chuyển vị (transposed convolution) để đầu ra là ảnh giả có kích thước lớn hơn so với đầu vào là nhiễu (noise) *z*.



Hình 7. Phép tích chập chuyển vị với bộ lọc có kích thước 2 x 2

2.2. Discriminator



Hình 8. Cấu trúc của mạng DCGAN Discriminator

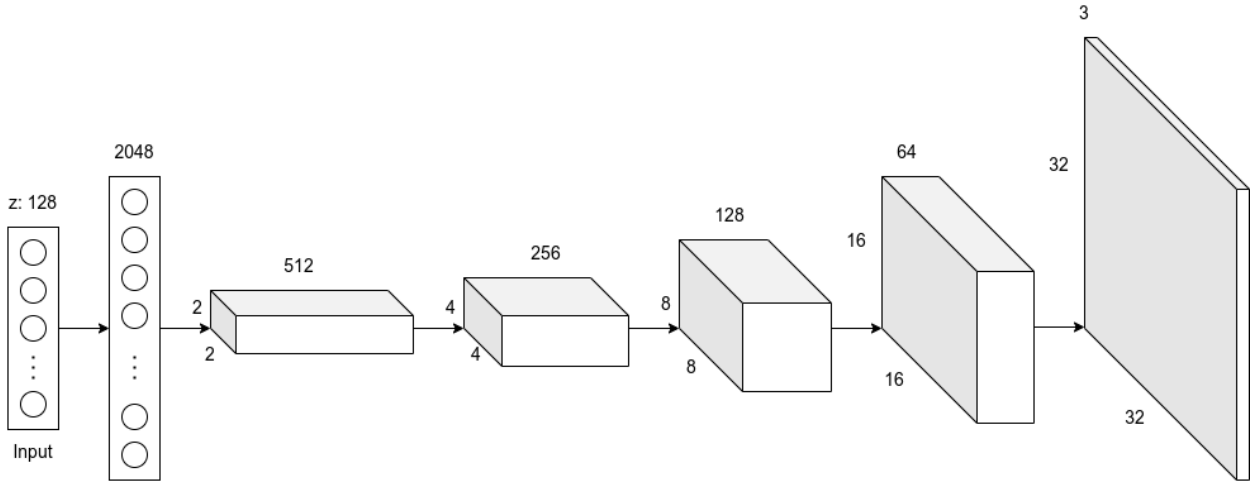
Trong cấu trúc Discriminator của DCGAN, ta sử dụng phép tích chập (convolution) để giảm kích thước ảnh sau đó kết nối fully-connected với 1 nơ-ron cuối cùng để giải bài toán phân loại nhị phân.

Chương 3. Lập trình và kết quả

3.1. Lập trình bài toán

a. Bộ cơ sở dữ liệu: MNIST, Fashion-MNIST, Cifar-10

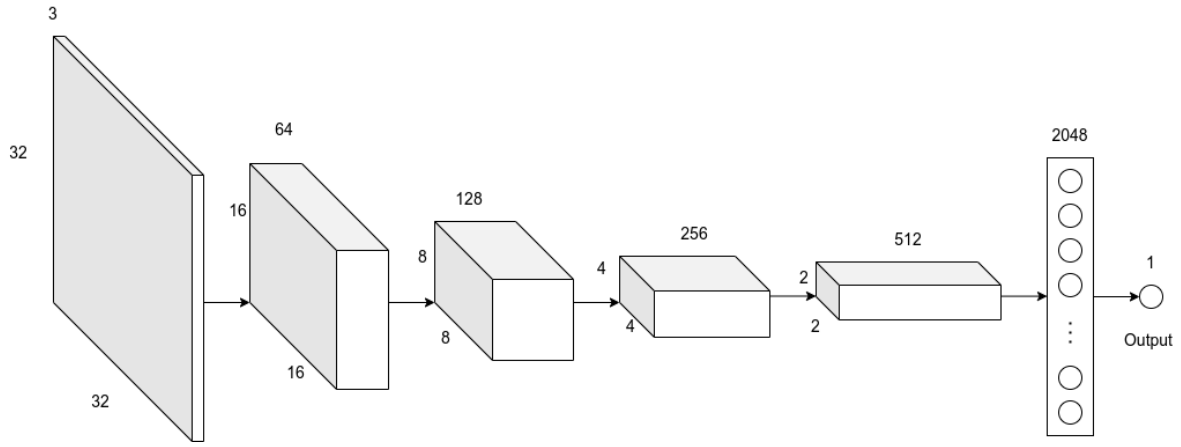
b. Cấu trúc Generator



Hình 9. Sơ đồ cấu trúc Generator DCGAN

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 2048)              206848
_________________________________________________________________
reshape_2 (Reshape)          (None, 2, 2, 512)         0
_________________________________________________________________
batch_normalization_6 (Batch (None, 2, 2, 512)         2048
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 2, 2, 512)         0
_________________________________________________________________
conv2d_transpose_5 (Conv2DTr (None, 4, 4, 256)         3277056
_________________________________________________________________
batch_normalization_7 (Batch (None, 4, 4, 256)         1024
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_transpose_6 (Conv2DTr (None, 8, 8, 128)         819328
_________________________________________________________________
batch_normalization_8 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_transpose_7 (Conv2DTr (None, 16, 16, 64)        204864
_________________________________________________________________
batch_normalization_9 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_transpose_8 (Conv2DTr (None, 32, 32, 3)         4803
_________________________________________________________________
batch_normalization_10 (Batc (None, 32, 32, 3)         12
_________________________________________________________________
activation_2 (Activation)    (None, 32, 32, 3)         0
=================================================================
Total params: 4,516,751
Trainable params: 4,514,825
Non-trainable params: 1,926
```

c. Cấu trúc Discriminator



Hình 10. Sơ đồ cấu trúc Discriminator DCGAN

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 16, 16, 64)        4864
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 128)         204928
_________________________________________________________________
batch_normalization_11 (Batc (None, 8, 8, 128)         512
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 256)         819456
_________________________________________________________________
batch_normalization_12 (Batc (None, 4, 4, 256)         1024
_________________________________________________________________
leaky_re_lu_11 (LeakyReLU)   (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 512)         3277312
_________________________________________________________________
batch_normalization_13 (Batc (None, 2, 2, 512)         2048
_________________________________________________________________
leaky_re_lu_12 (LeakyReLU)   (None, 2, 2, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 2049
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0
=================================================================
Total params: 4,312,193
Trainable params: 4,310,401
Non-trainable params: 1,792
```

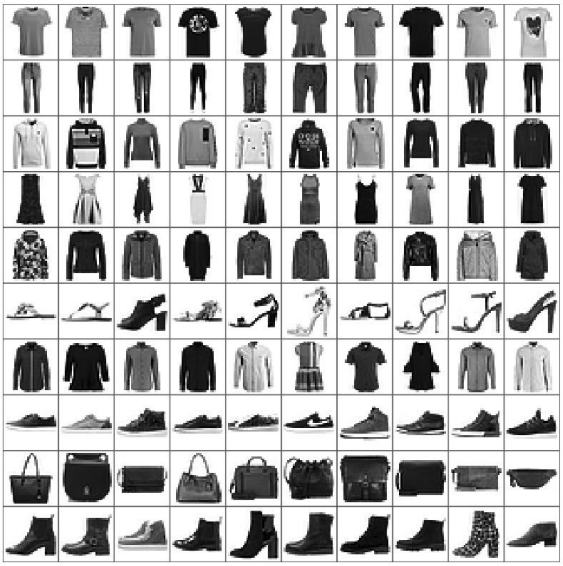
d. Siêu tham số và phương pháp tối ưu:
- Tốc độ học: $\alpha = 0.0002$.
- Tối ưu: Adam.

3.2. Kết quả thực nghiệm

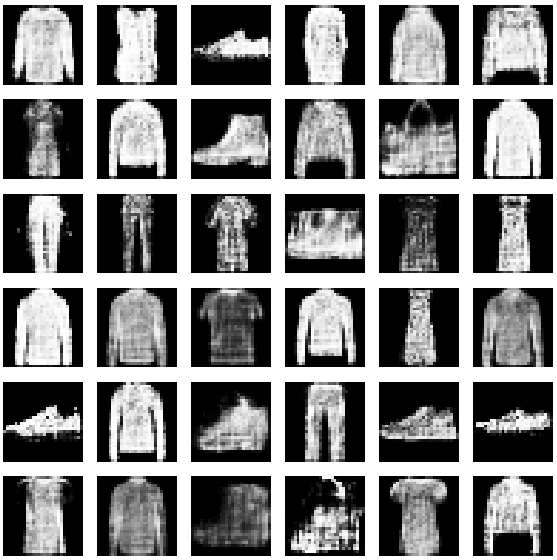
a. Trên cơ sở dữ liệu MNIST:

Nhận xét: mạng DCGAN cho kết quả tốt hơn so với GAN.

b. Trên cơ sở dữ liệu Fashion-MNIST



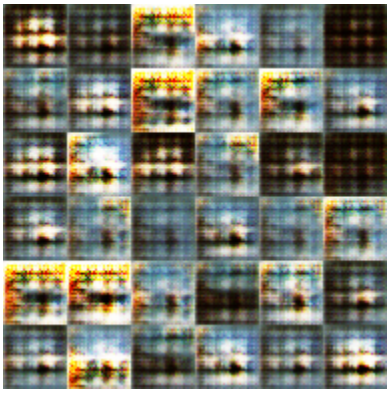
Groundtruth



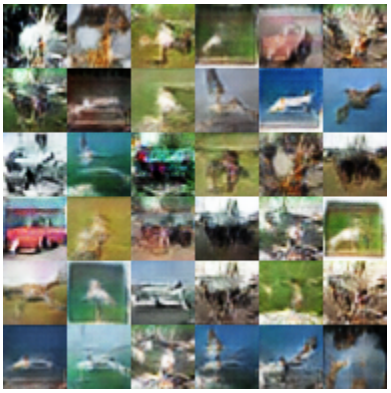
DCGAN

c. Trên cơ sở dữ liệu Cifar-10

Groundtruth



DCGAN (20 epochs)



DCGAN (200 epochs)

Nhận xét: Ở những epoch đầu tiên thì Generator chỉ sinh ra noise, tuy nhiên khi huấn luyện lâu hơn ta thấy mạng đã có thể học được các thuộc tính của ảnh trong tập dữ liệu.

Danh mục tài liệu tham khảo

1. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. "*Generative Adversarial Networks*". arXiv:1406.2661 2014.
2. Alec Radford, Luke Metz, Soumith Chintala. "*Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*". arXiv:1511.06434 2016.
3. Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. "*Dive into Deep Learning*". URL https://d2l.ai/d2l-en.pdf