Trung tâm đào tạo quốc tế (ITEC)

Đại học Khoa học tự nhiên TP HCM

CS300-Artificial Intelligence

BÁO CÁO ĐỒ ÁN

THỰC HÀNH CUỐI KÌ

Giảng viên hướng dẫn

ThS. Nguyễn Hải Đăng

ThS. Nguyễn Thành An

Thông tin nhóm:

Đoàn Văn Thanh Liêm – MSSV: 1859027

Đỗ Phương Nhật Minh – MSSV: 1859032

Trần Gia Hòa – MSSV: 1859016

Đoàn Minh Tuấn – MSSV: 1859048

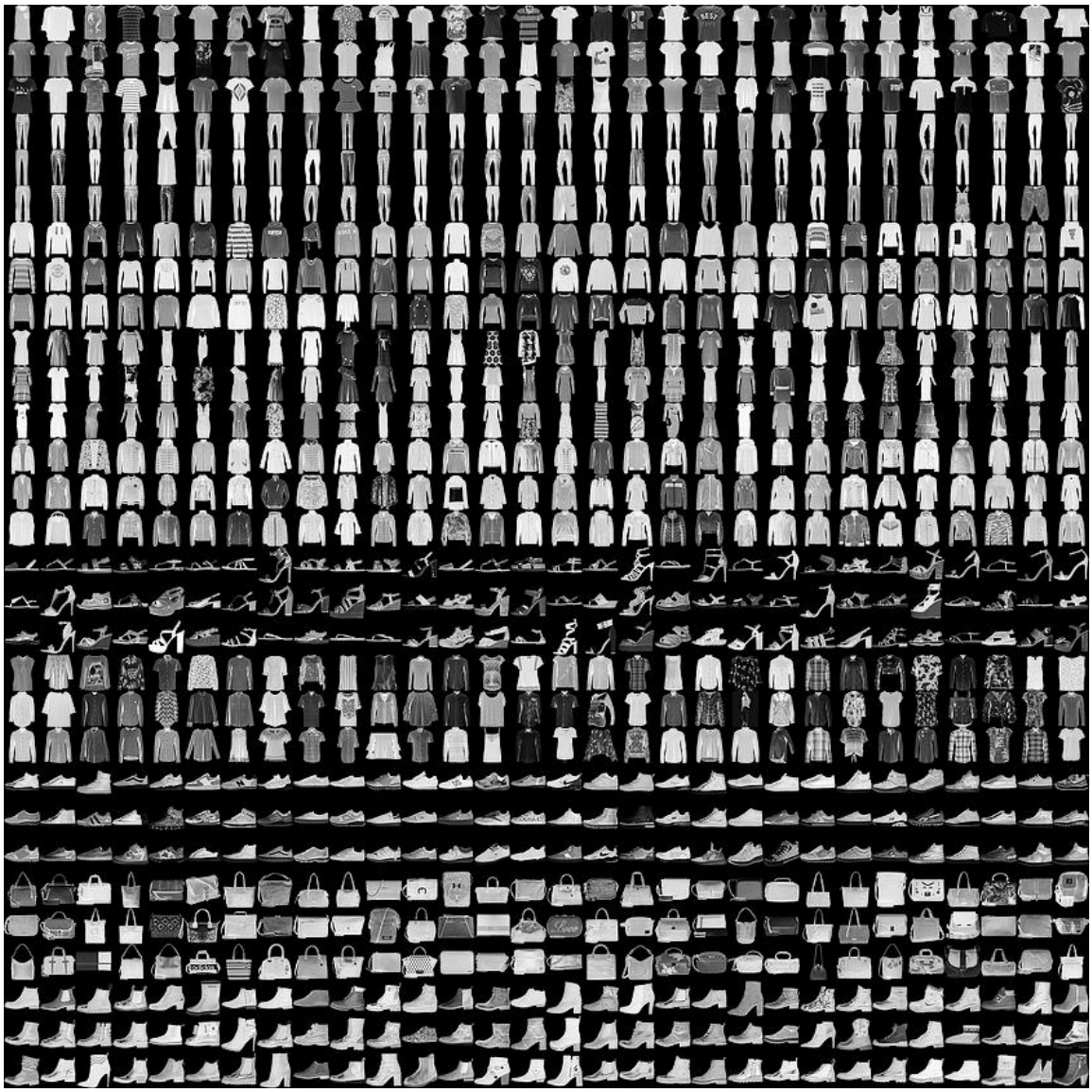
MỤC LỤC

I. Sơ lược về đồ án

1.1. Yêu cầu bài toán

– Trong đồ án cuối kỳ, yêu cầu được đặt ra là sử dụng tập dữ liệu Fashion MNIST được cung cấp sẵn để lựa chọn và xây dựng hai mô hình nhận dạng hình ảnh riêng biệt nhau. Trong đó:

- Input: ảnh (28x28) pixels (ảnh grayscale)
- Output: loại trang phục (0-9)



– Sau khi thực thi huấn luyện và có được kết quả thực nghiệm cần đưa ra nhận xét về mỗi mô hình, so sánh hiệu quả giữa hai mô mình đã được lựa chọn và phương pháp cải tiến những mô hình trên để đạt kết quả nhận diện tốt hơn.

2.2. Dữ liệu huấn luyện

– Fashion-MNIST là một tập dữ liệu về hình ảnh của Zalando — bao gồm một tập huấn luyện gồm 60.000 ví dụ và một tập kiểm tra 10.000 ví dụ. Mỗi ví dụ là một hình ảnh grayscale 28x28, được liên kết với một nhãn từ 10 lớp.

– Mỗi ví dụ đào tạo và kiểm tra được gán cho một trong các nhãn sau:

Nhãn	Giải thích
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

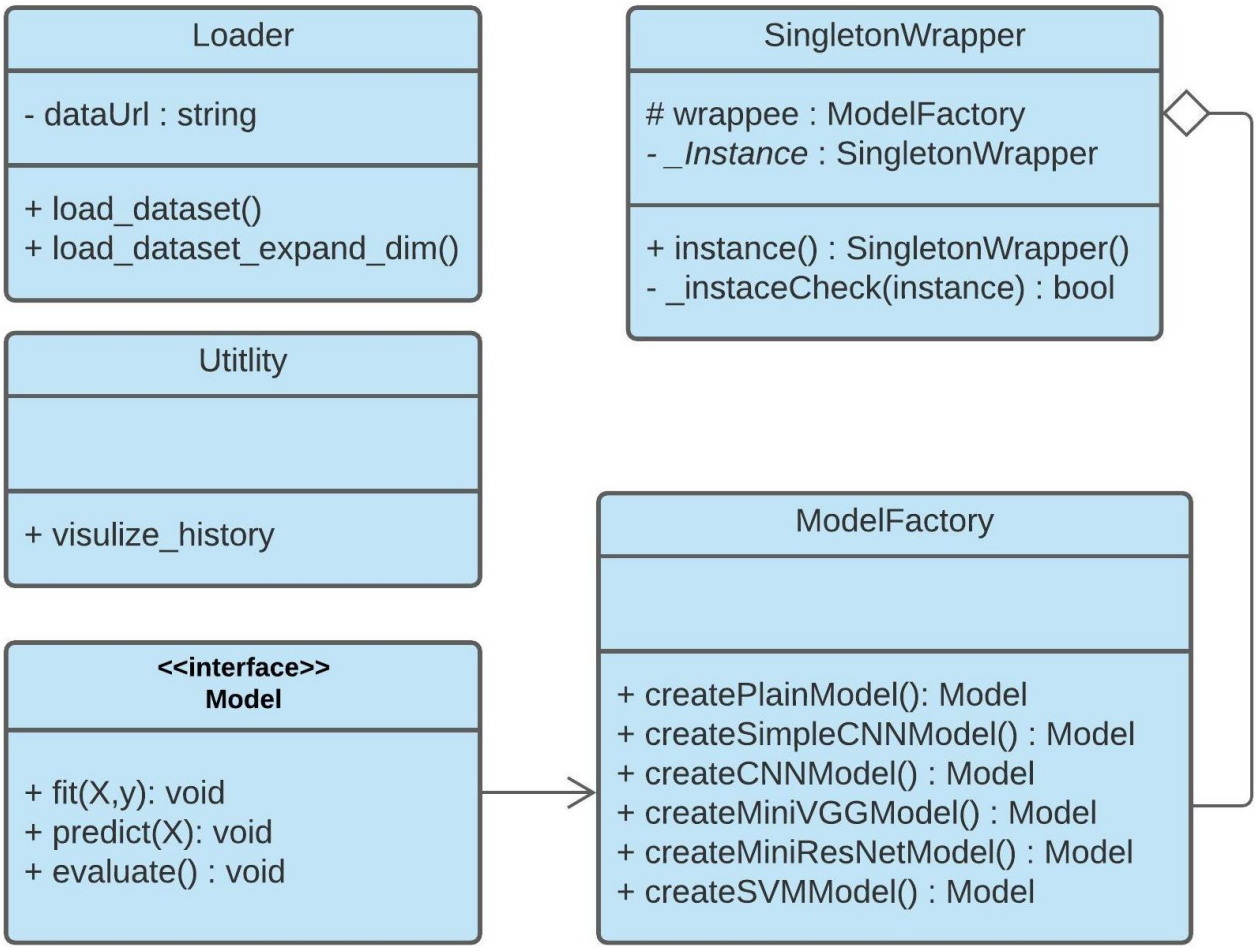
II. Bảng phân công nhiệm vụ thực hiện

MSSV	Họ Tên	% đóng góp	Công việc thực hiện
1859027	Đoàn Văn Thanh Liêm	25%	Giải quyết bài toán bằng việc sử dụng mô hình ***Mạng nơ-ron tích chập - Convolutional Neural Network*** và viết tài liệu bổ trợ.
1859048	Đoàn Minh Tuấn	25%	
1859032	Đỗ Phương Nhật Minh	25%	Giải quyết bài toán bằng phương pháp ***Máy vector hỗ trợ - Support Vector Machine (SVM)*** và viết tài liệu bổ trợ.
1859016	Trần Gia Hòa	25%	

III. Các phương pháp giải quyết bài toán

3.0. Kiến trúc mã nguồn đồ án

– Nhằm giải quyết việc kiến trúc và sử dụng các mô hình học máy một cách hiệu quả và tái kiến trúc dễ dàng, đồ án được cấu trúc theo hướng đối tượng theo mô hình UML như sau:



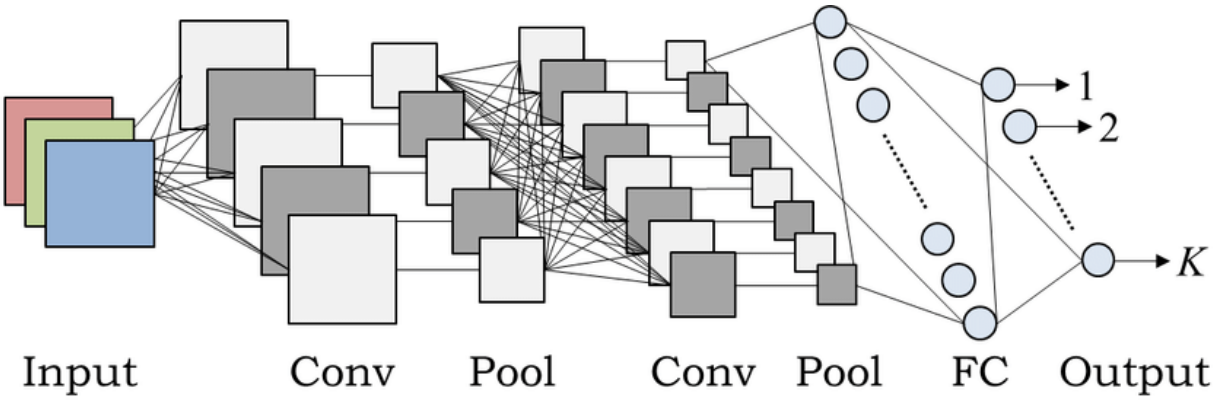
– Trong đó, việc khởi tạo và truy cập đến các mô hình máy học được quản lý bởi **ModelFactory** (áp dụng mẫu kiến trúc hướng đối tượng *Abstract Factory và Factory Method*). Bên cạnh đó, nhằm mục đích chỗ khởi tạo duy nhất một thể hiện của ModelFactory và khiến nó có tầm vực toàn cục (global scope), **ModelFactory** được bao bọc bởi một **SingletonWrapper** (áp dụng mẫu kiến trúc *Singleton* và ý tưởng củ mẩu *Decorator*)

– Các tác vụ hỗ trợ việc tải các dữ liệu và xữ lý đồ họa của các thông tin trong tập huấn luyện và kiểm tra được đối tượng hóa thành hai lớp **Loader** và **Utitlity**

3.1. Mạng nơ-ron tích chập - Convolutional Neural Network

3.1.1. Khái niệm chung về mạng tích chập

– Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

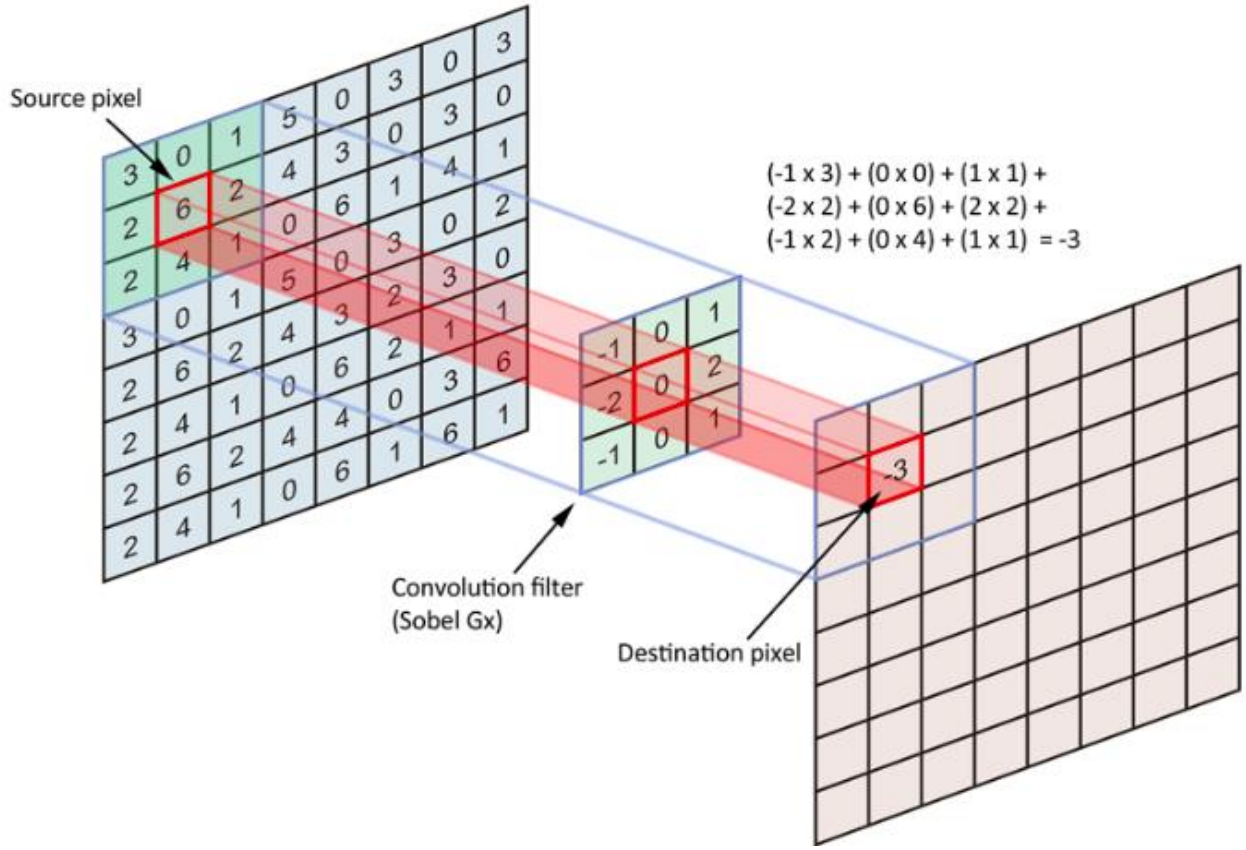


– Tích chập là ứng dụng đơn giản của bộ lọc đối với đầu vào dẫn đến kích hoạt. Việc áp dụng lặp lại cùng một bộ lọc cho một đầu vào dẫn đến việc tạo ra một dãy tín hiệu được gọi là ma trận lớp đầu vào (feature map), cho biết vị trí và độ mạnh của đặc điểm được phát hiện trong đầu vào.

– Sự đổi mới của mạng nơ-ron tích chập so với các mạng nơ-ron truyền thống là khả năng tự động học một số lượng lớn các bộ lọc song song cụ thể cho một tập dữ liệu huấn luyện dưới các ràng buộc của một vấn đề mô hình dự đoán cụ thể, chẳng hạn như phân loại hình ảnh. Kết quả là mô hình có thể phát hiện ra được các đặc điểm có tính cụ thể cao có thể được phát hiện ở bất kỳ đâu trên hình ảnh đầu vào.

3.1.2. Khái niệm các lớp lọc (filter hay kernel)

– Trong mạng nơ-ron tích chập, bộ lọc(filter) phát hiện các mẫu không gian, chẳng hạn như các cạnh trong hình ảnh bằng cách phát hiện những thay đổi trong giá trị cường độ của hình ảnh. Trong mạng nơ-ron tích chập sử dụng bộ lọc(filter) để trích xuất một số 'đặc điểm' từ hình ảnh đầu vào

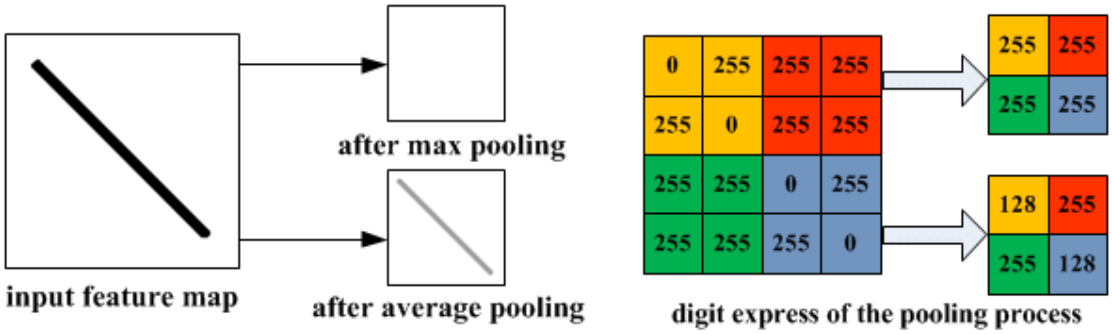


– Bộ lọc được biểu diễn bằng một vectơ trọng số mà mô hình dùng để tích chập với hình ảnh đầu vào. Bộ lọc, tương tự như bộ lọc gặp phải trong quá trình xử lý tín hiệu kĩ thuật số, cung cấp cho chúng ta một thước đo về việc một phần của bức ảnh đầu vào giống với một đặc điểm. Một đặc điểm này có thể là cạnh thẳng đứng hoặc hình vòm.

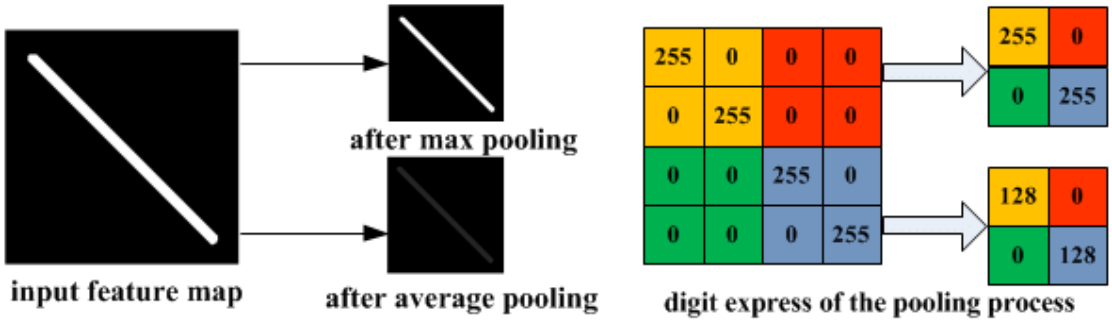
3.1.3. Khái niệm các lớp Max pooling và Average pooling

– Max pooling là một loại lớp thường được thêm vào CNN sau các lớp phức hợp riêng lẻ.
– Khi được thêm vào một mô hình, Max pooling làm giảm kích thước của hình ảnh bằng cách giảm số lượng pixel trong đầu ra từ lớp phức hợp trước đó.
– Max pooling làm giảm độ phân giải của đầu ra nhất định của một lớp phức hợp, mạng sẽ xem xét các khu vực lớn hơn của hình ảnh tại một thời điểm trong tương lai, điều này làm giảm lượng tham số trong mạng và do đó giảm tải tính toán.
– Ngoài ra, max pooling cũng có thể giúp giảm tình trạng quá tải. Trực giác về lý do tại sao tính năng gộp tối đa hoạt động là đối với một hình ảnh cụ thể, mạng của chúng tôi sẽ tìm cách trích xuất một số tính năng cụ thể.

– Average pooling là một kiểu tổng hợp khác và đó là nơi bạn lấy giá trị trung bình từ mỗi vùng thay vì giá trị tối đa. [1]



(a) Illustration of max pooling drawback



(b) Illustration of average pooling drawback

– Hiện tại, max pooling được sử dụng nhiều hơn average pooling

3.1.4. Khái niệm Cross-Entropy

– Cross entropy giữa hai phân phối **p** và **q** được định nghĩa là:

$$H(\mathbf{p}, \mathbf{q}) = \mathbf{E}_{\mathbf{p}}[-\log \mathbf{q}]$$

– Với **p** và **q** là rời rạc (như **y** và **a** trong bài toán của chúng ta), công thức này được viết dưới dạng: [2]

$$H(\mathbf{p}, \mathbf{q}) = -\sum_{i=1}^{C} p_i \log q_i \quad (1)$$

– Có hai nhận xét quan trọng sau đây:

- Giá trị nhỏ nhất của cả hai hàm số đạt được khi **q**=**p** tại hoành độ của các điểm màu xanh lục.
- Quan trọng hơn, hàm cross entropy nhận giá trị rất cao (tức loss rất cao) khi **q** ở xa **p**. Trong khi đó, sự chênh lệch giữa các loss ở gần hay xa nghiệm của hàm bình phương khoảng cách **(q−p)2** là không đáng kể. Về mặt tối ưu, hàm cross entropy sẽ cho nghiệm *gần* với **p** hơn vì những nghiệm ở xa bị *trừng phạt* rất nặng.

– Hai tính chất trên đây khiến cho cross entropy được sử dụng rộng rãi khi tính khoảng cách giữa hai phân phối xác suất.

$$J = -\sum_{i=1}^{N} y_i \log(h_\theta(x_i)) + (1 - y_i) \log(1 - h_\theta(x_i))$$

3.1.5. Khái niệm Batch Normalization

– Batch Normalization là một kỹ thuật để đào tạo các mạng nơ-ron rất sâu, chuẩn hóa các đầu vào thành một lớp cho mỗi lô nhỏ. Điều này có tác dụng ổn định quá trình học và giảm đáng kể vòng lặp huấn luyện cần thiết để đào tạo mạng sâu (deep networks). [3]

3.1.6. Non-Linear : RELU

– ReLU (Rectified Linear Units, f = max(0, x)) là hàm kích hoạt phổ biến nhất cho CNN tại thời điểm của bài viết, được giới thiệu bởi Geoffrey E. Hinton năm 2010. Trước khi hàm ReLU được áp dụng thì những hàm như sigmoid hay tanh mới là những hàm được sử dụng phổ biến. Hàm ReLU được ưa chuộng vì tính toán đơn giản, giúp hạn chế tình trạng vanishing gradient, và cũng cho kết quả tốt hơn. ReLU cũng như những hàm kích hoạt khác, được đặt ngay sau tầng convolution, ReLU sẽ gán những giá trị âm bằng 0 và giữ nguyên giá trị của đầu vào khi lớn hơn 0. [4]

– ReLU cũng có một số vấn đề tiềm ẩn như không có đạo hàm tại điểm 0, giá trị của hàm ReLU có thể lớn đến vô cùng và nếu chúng ta không khởi tạo trọng số cẩn thận, hoặc khởi tạo learning rate quá lớn thì những neuron ở tầng này sẽ rơi vào trạng thái chết, tức là luôn có giá trị < 0. [4]

3.1.7. Lý do chọn mô hình mạng nơ-ron tích trập

- Lý do chính khiến tôi chọn kiến trúc mô hình mạng nơ-ron tích trập là vì nó là một trong những phương pháp phân loại hình ảnh tân tiến nhất trong lĩnh vực thị giác máy tính.
- Đối với thiết kế kiến trúc các lớp trong mô hình học máy, tôi tham khảo mô hình mạng nơ-ron sâu là VGG19 (Very Deep Convolutional Networks for Large-Scale Image Recognition) [5]

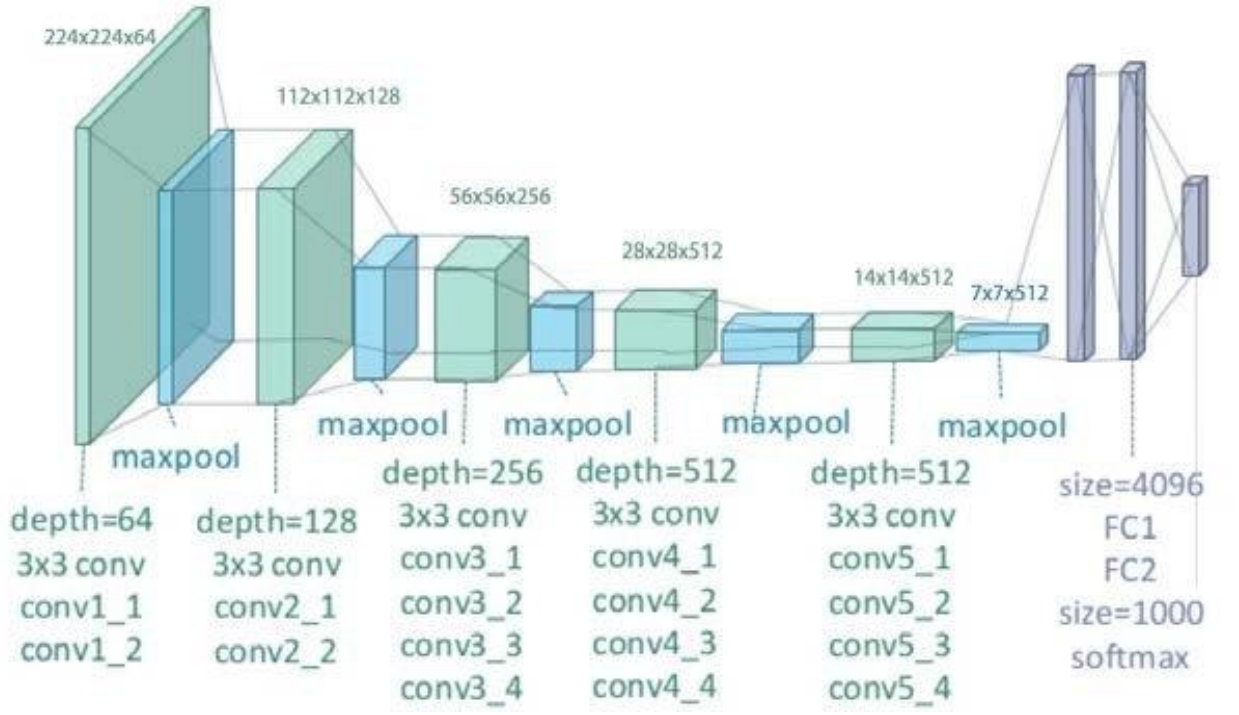


Figure 1. VGG19 Archotecture

3.2. Máy vector hỗ trợ - Support Vector Machine (SVM)

3.2.1. Khái niệm chung

- SVM (Support Vector Machine) là một thuật toán học máy có giám sát được sử dụng rất phổ biến ngày nay trong các bài toán phân lớp (classification) hay hồi qui (Regression). Ở bài toán chúng ta, ta chỉ xét đến ứng dụng SVM cho các bài toán phân lớp (classification).

3.2.2. Khoảng cách từ 1 điểm đến 1 mặt phẳng (siêu mặt phẳng)

- Trong không gian 2 chiều, ta biết rằng khoảng cách từ một điểm có toạ độ (x0,y0)(x0,y0) tới đường thẳng có phương trình w1x+w2y+b=0 được xác định bởi:

$$\frac{|w_1x_0 + w_2y_0 + b|}{\sqrt{w_1^2 + w_2^2}}$$

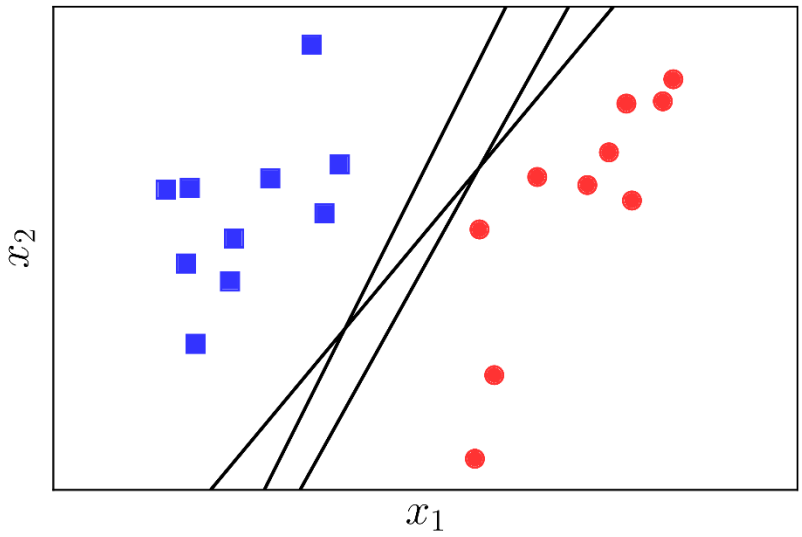
- Trong không gian ba chiều, khoảng cách từ một điểm có toạ độ (x0,y0,z0) tới một mặt phẳng có phương trình w1x+w2y+w3z+b=0 được xác định bởi:

$$\frac{|w_1x_0 + w_2y_0 + w_3z_0 + b|}{\sqrt{w_1^2 + w_2^2 + w_3^2}}$$

- Hơn nữa, nếu ta bỏ dấu trị tuyệt đối ở tử số, chúng ta có thể xác định được điểm đó nằm về phía nào của *đường thẳng* hay *mặt phẳng* đang xét. Những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu dương nằm về cùng 1 phía (tôi tạm gọi đây là *phía dương* của đường thẳng), những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu âm nằm về phía còn lại (tôi gọ là *phía âm*). Những điểm nằm trên *đường thẳng/măt phẳng* sẽ làm cho tử số có giá trị bằng 0, tức khoảng cách bằng 0.

3.2.3. Về bài toán phân chia 2 class

- Giả sử rằng có hai class khác nhau được mô tả bởi các điểm trong không gian nhiều chiều, hai classes này *linearly separable*, tức tồn tại một siêu phẳng phân chia chính xác hai classes đó. Hãy tìm một siêu mặt phẳng phân chia hai classes đó, tức tất cả các điểm thuộc một class nằm về cùng một phía của siêu mặt phẳng đó và ngược phía với toàn bộ các điểm thuộc class còn lại.

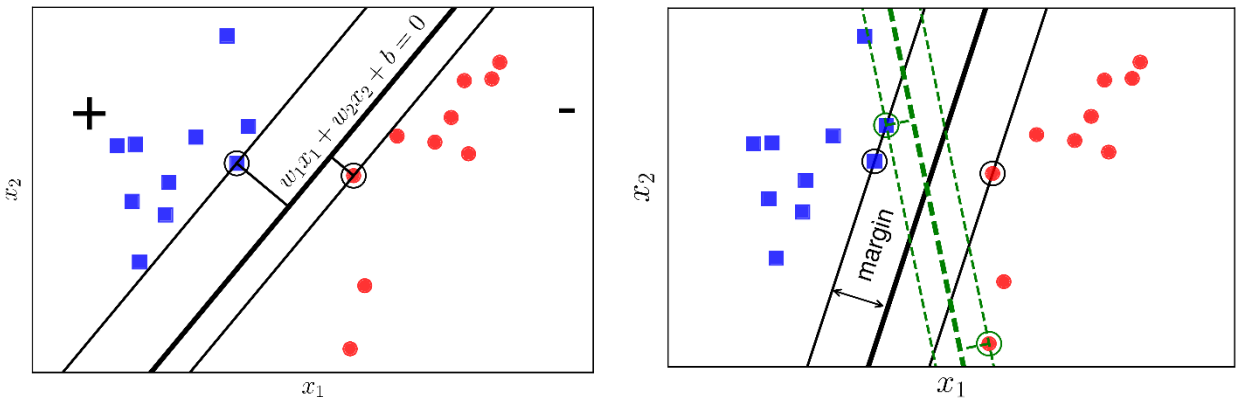


Hình 2: Các mặt phân cách hai classes linearly separable.

(Nguồn: https://machinelearningcoban.com/2017/04/09/smv/)

– Câu hỏi đặt ra là: trong vô số các mặt phân chia đó, đâu là mặt phân chia tốt nhất *theo một tiêu chuẩn nào đó*? Trong ba đường thẳng minh họa trong Hình 1 phía trên, có hai đường thẳng khá *lệch* về phía class hình tròn đỏ. Điều này có thể khiến cho lớp màu đỏ *không vui vì lãnh thổ xem ra bị lấn nhiều quá*. Liệu có cách nào để tìm được đường phân chia mà cả hai classes đều cảm thấy *công bằng* và *hạnh phúc* nhất hay không?

– Chúng ta cần tìm một tiêu chuẩn để đo sự *hạnh phúc* của mỗi class. Hãy xem Hình 2 dưới đây



Hình 3: Margin của hai classes là bằng nhau và lớn nhất có thể.
(Nguồn: https://machinelearningcoban.com/2017/04/09/smv/)

– Nếu ta định nghĩa *mức độ hạnh phúc* của một class tỉ lệ thuận với khoảng cách gần nhất từ một điểm của class đó tới đường/mặt phân chia, thì ở Hình 2 trái, class tròn đỏ sẽ

không được hạnh phúc cho lắm vì đường phân chia gần nó hơn class vuông xanh rất nhiều. Chúng ta cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi class (các điểm được khoanh tròn) tới đường phân chia là như nhau, như thế thì mới *công bằng*. Khoảng cách như nhau này được gọi là *margin* (*lề*).

- Đã có *công bằng* rồi, chúng ta cần *văn minh* nữa. *Công bằng* mà cả hai đều *kém hạnh phúc như nhau* thì chưa phải là *văn mình* cho lắm.
- Chúng ta xét tiếp Hình 2 bên phải khi khoảng cách từ đường phân chia tới các điểm gần nhất của mỗi class là như nhau. Xét hai cách phân chia bởi đường nét liền màu đen và đường nét đứt màu lục, đường nào sẽ làm cho cả hai class *hạnh phúc hơn*? Rõ ràng đó phải là đường nét liền màu đen vì nó tạo ra một *margin* rộng hơn.
- Việc *margin* rộng hơn sẽ mang lại hiệu ứng phân lớp tốt hơn vì *sự phân chia giữa hai classes là rạch ròi hơn*. Việc này, sau này các bạn sẽ thấy, là một điểm khá quan trọng giúp *Support Vector Machine* mang lại kết quả phân loại tốt hơn so với *Neural Network với 1 layer*, tức Perceptron Learning Algorithm.
- Bài toán tối ưu trong *Support Vector Machine* (SVM) chính là bài toán đi tìm đường phân chia sao cho *margin* là lớn nhất. Đây cũng là lý do vì sao SVM còn được gọi là *Maximum Margin Classifier*. Nguồn gốc của tên Support Vector Machine sẽ sớm được làm sáng tỏ.

3.2.4. Lý do chọn SVM

- Ta chọn SVM để giải bài toán vì trên trang scikit-learn.org cho ta thấy một ví dụ về nhận diện chữ số viết tay (Recognizing hand-written digits). Dựa vào mô tả ví dụ bài toán, ta thấy rằng những bức ảnh 8x8 pixel về các chữ số viết tay riêng lẻ trong khoảng từ 0 đến 9 được sử dụng như là dataset.
- Với việc coi bài toán này là bài toán phân lớp (classification) thì ta thấy rằng tập hợp các nhãn (labels) sẽ một chữ số trong khoảng từ 0 đến 9, tức chúng ta có 10 nhãn (labels)Tương tự, bài toán phân loại phụ kiện thời trang của chúng ta cũng có tập hợp nhãn gồm 10 phần tử .Việc xử lí hình ảnh cũng sẽ được thực hiện giống như ví dụ của SVM, các bức ảnh 28x28 pixel (được lưu bằng mảng 12 chiều) sẽ được phẳng hóa thành 1 mảng một chiều có 784 phần tử.

IV. Thực nghiệm

4.1. Mạng nơ-ron tích chập - Convolutional Neural Network

4.1.1. Tiền xử lý dữ liệu huấn luyện

– Chuẩn hóa min-max là phương pháp đơn giản nhất trong việc co giãn phạm vi của đặc trưng bằng việc co giãn chúng về phạm vi [0,1] hoặc [-1,1]. Công thức chung được cho như sau:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

– Với x là giá trị ban đầu, x_{scaled} là giá trị sau khi chuẩn hóa, x_{min} là giá trị nhỏ nhất và x_{max} là giá trị lớn nhất của đặc trưng. Vì vậy, dữ liệu đầu vào sẽ được chuẩn hóa lại như sau

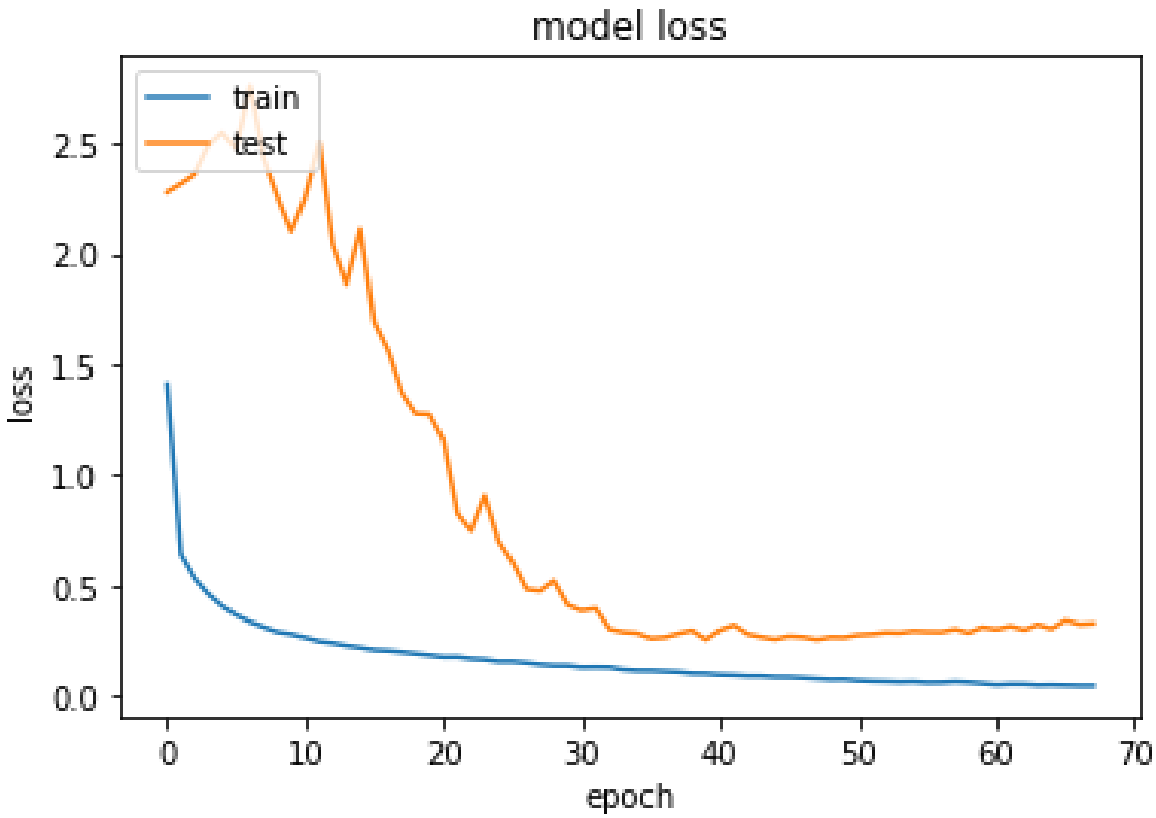
```
You, 2 days ago • Init
# Rescale the images from [0,255] to the [0.0, 1.0] range.
x_train_1, x_test_1 = np.array(
    x_train, dtype=np.float32)/255.0, np.array(x_test, dtype=np.float32)/255.0
y_train_1, y_test_1 = tf.keras.utils.to_categorical(
    y_train, 10, dtype=np.uint8), tf.keras.utils.to_categorical(y_test, 10, dtype=np.uint8)
```

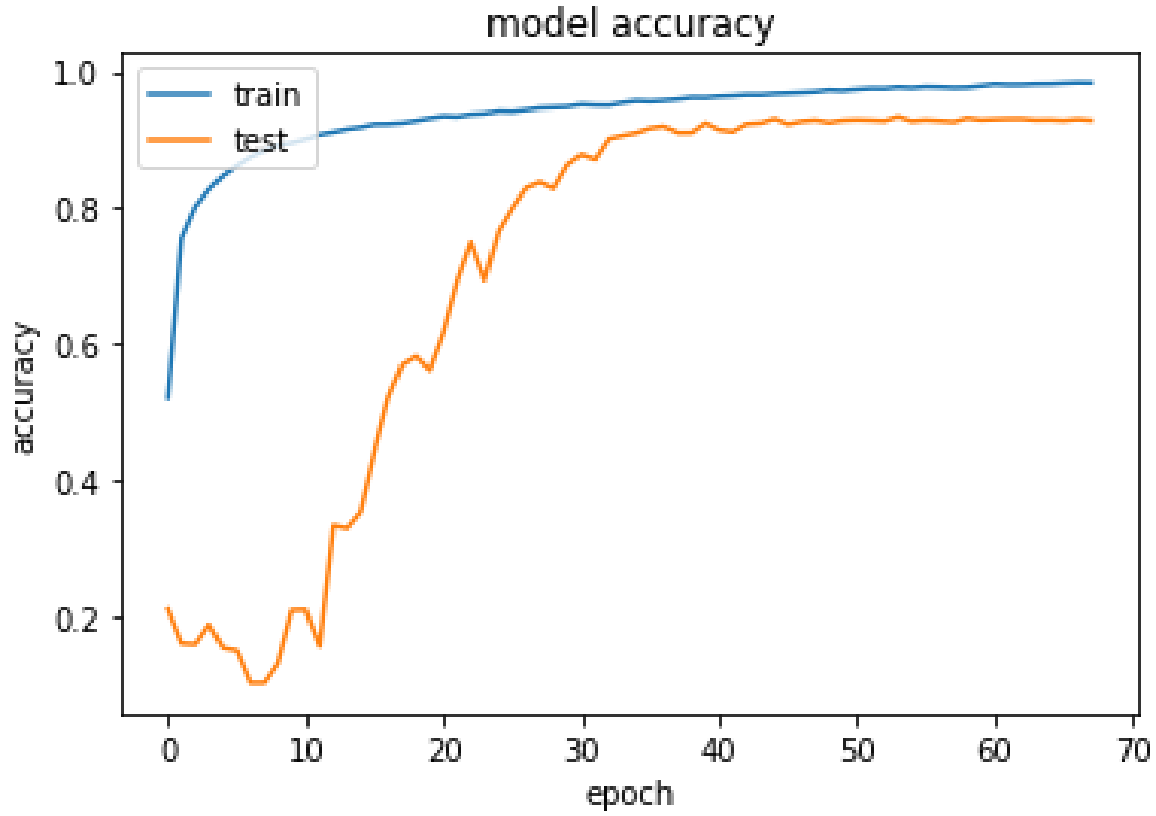
4.1.2. Mô hình huấn luyện

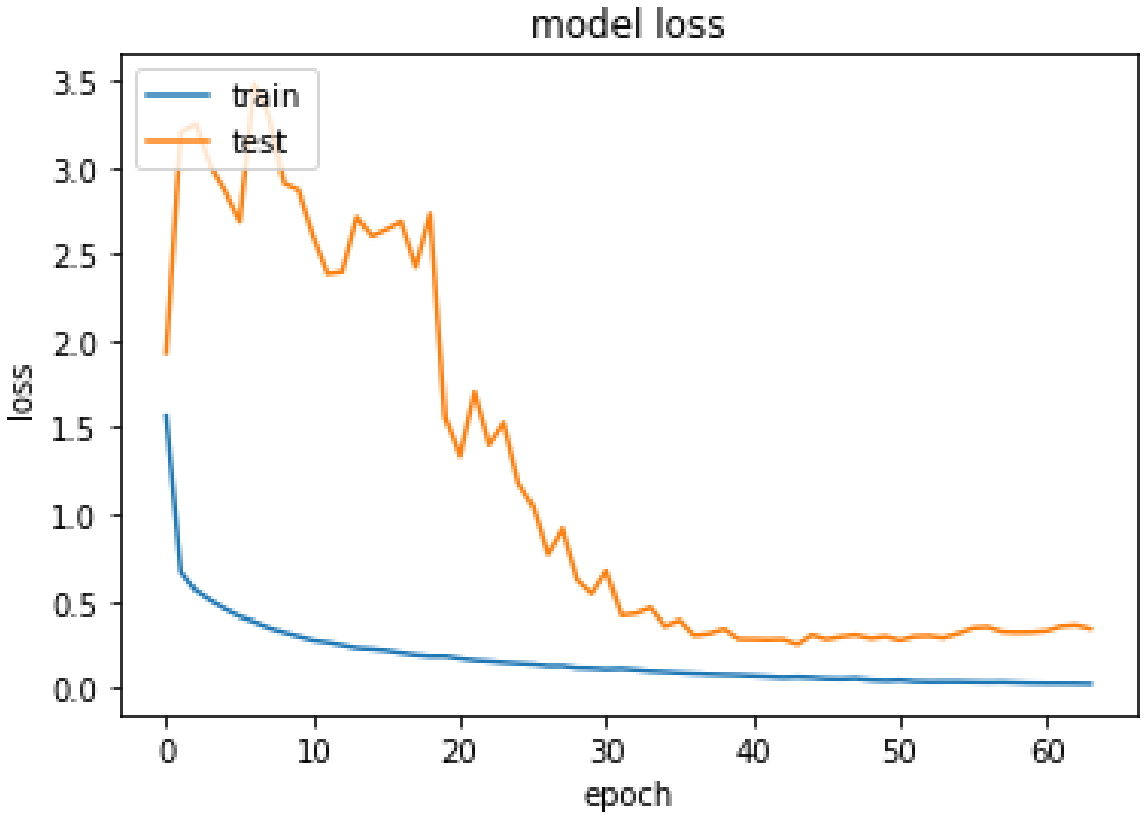
Layer (type)	Output Shape	Param#
Up_sampling3d_4(upsampling3d)	(None, 28, 28, 1)	0
Conv2d_48 (conv2d)	(None, 28, 28, 32)	320
Batch_normalization_56 (batch normalization)	(None, 28, 28, 32)	112
Conv2d_49 (conv2d)	(None, 28, 28, 32)	9248
Batch_normalization_57 (batch normalization)	(None, 28, 28, 32)	112
Max_pooling2d_16 (max pooling)	(None, 14, 14, 32)	0
Dropout_16 (dropout)	(None, 14, 14, 32)	0
Conv2d_50 (conv2d)	(None, 14, 14, 64)	18496
Batch_normalization_58 (batch normalization)	(None, 14, 14, 64)	56
Conv2d_51 (conv2d)	(None, 14, 14, 64)	36928
Batch_normalization_59 (batch normalization)	(None, 14, 14, 64)	56
Max_pooling2d_17 (max pooling)	(None, 7, 7, 64)	0
Dropout_17 (dropout)	(None, 7, 7, 64)	0
Conv2d_52 (conv2d)	(None, 7, 7, 128)	73856
Batch_normalization_60 (batch normalization)	(None, 7, 7, 128)	28
Conv2d_53 (conv2d)	(None, 7, 7, 128)	147584
Batch_normalization_61 (batch normalization)	(None, 7, 7, 128)	28
Conv2d_54 (conv2d)	(None, 7, 7, 128)	147584
Batch_normalization_62 (batch normalization)	(None, 7, 7, 128)	28
Conv2d_55 (conv2d)	(None, 7, 7, 128)	147584
Batch_normalization_63 (batch normalization)	(None, 7, 7, 128)	28
Max_pooling2d_18 (max pooling)	(None, 3, 3, 128)	0
Dropout_18 (dropout)	(None, 3, 3, 128)	0
Conv2d_56 (conv2d)	(None, 3, 3, 256)	295168
Batch_normalization_64 (batch normalization)	(None, 3, 3, 256)	12
Conv2d_57 (conv2d)	(None, 3, 3, 256)	590080
Batch_normalization_65 (batch normalization)	(None, 3, 3, 256)	12
Conv2d_58 (conv2d)	(None, 3, 3, 256)	590080
Batch_normalization_66 (batch normalization)	(None, 3, 3, 256)	12
Conv2d_59 (conv2d)	(None, 3, 3, 256)	590080
Batch_normalization_67 (batch normalization)	(None, 3, 3, 256)	12
Max_pooling2d_19 (max pooling)	(None, 1, 1, 256)	0
Dropout_19 (dropout)	(None, 1, 1, 256)	0
Flatten_4 (flatten)	(None, 256)	0
Dense_12 (dense)	(None, 256)	65792
Batch_normalization_68 (batch normalization)	(None, 256)	1024
Dense_13 (dense)	(None, 256)	65792
Batch_normalization_69 (batch normalization)	(None, 256)	1024
Dense_14 (dense)	(None, 10)	2570
Total params:		**2783706**
Trainable params:		**2782434**
Non-trainable params:		**1272**

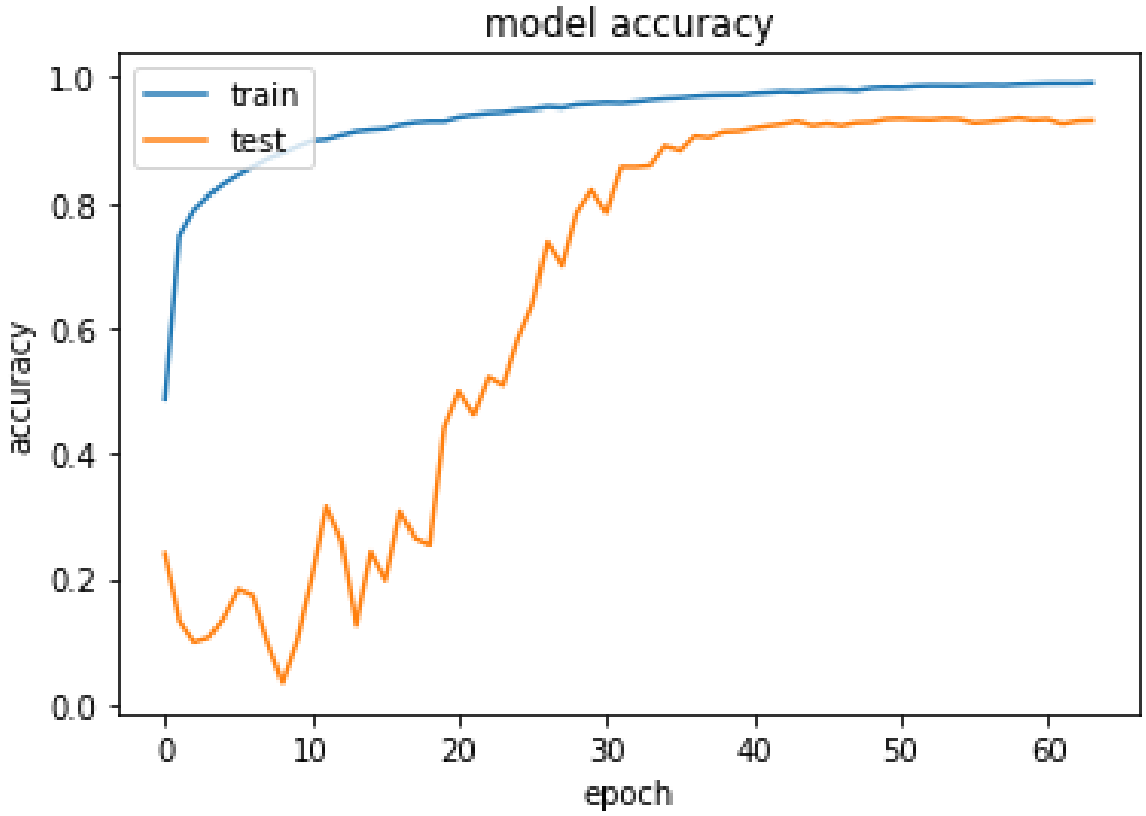
4.1.3. Kết quả sơ bộ

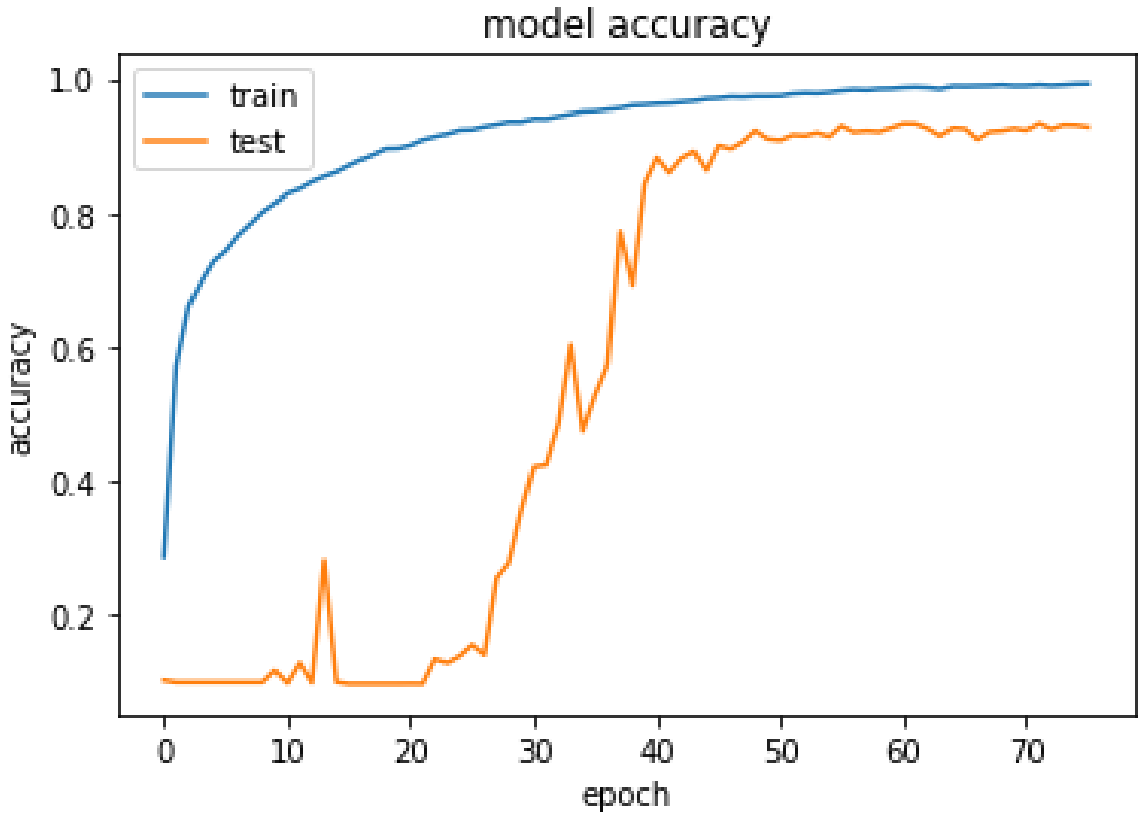
ID	Enviromemt	Time	Batchsize	Epochs	TrainAcc	ValAcc
0.1	GPU	4527	1024	**49/200**	**97.24%**	**92.98%**
0.2		4659		51/200	97.67%	92.45%
0.3		4431		48/200	97.13%	92.87%
Dùng TPU để xử lý việc huấn luyện						
1.1	TPU	69 giây	4096	**68/200**	**98.24%**	**93.91%**
1.2		71 giây		71/200	98.67%	93.78 %
1.3		61 giây		63/200	98.13%	93.83%
2.1		82 giây	2048	53/200	98.23%	93.80%
2.2		87 giây		55/200	98.09%	93.24%
2.3		78 giây		**49/200**	**98.1%**	**93.54%**
2.1		89 giây	1024	39/200	97.25%	92.98%
2.2		91 giây		**37/200**	**97.87%**	**93.11%**
2.3		99 giây		43/200	98.02%	93.02%
2.1		124 giây	512	**41/200**	**97.88%**	**93.75%**
2.2		131 giây		46/200	97.39%	92.12%
2.3		121 giây		60/200	97.79%	92.41%
Dùng Data Augmentation để tăng training data lên 600000 ảnh						
3.1	TPU	321 giây	2048	**32/200**	**97.67%**	**94%**
3.2		334 giây		33/200	97.98%	94%
3.3		315 giây		31/200	97.45%	93.98%

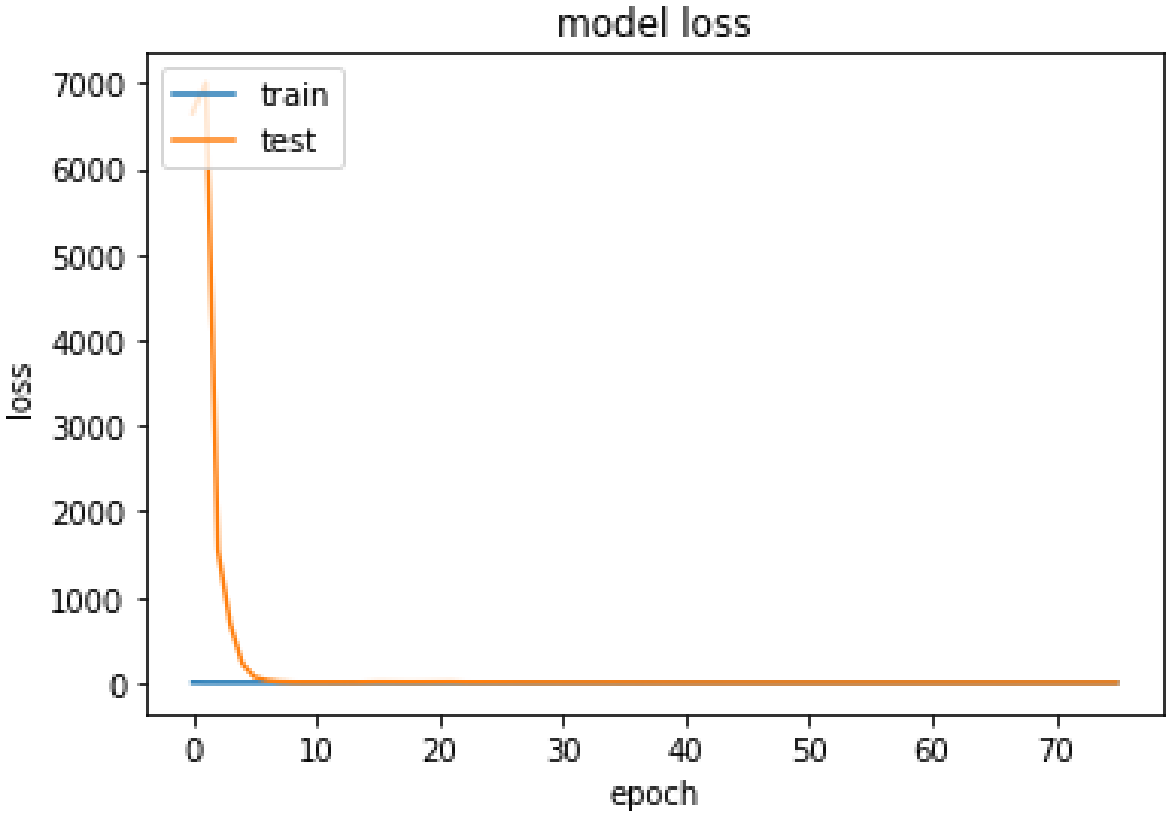

model loss
train
test
loss
epoch
0.0
0.5
1.0
1.5
2.0
2.5
0
10
20
30
40
50
60
70


model accuracy
train
test
accuracy
epoch
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
70


model loss
train
test
loss
epoch
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
10
20
30
40
50
60


model accuracy
train
test
accuracy
epoch
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60


model accuracy
train
test
accuracy
epoch
1.0
0.8
0.6
0.4
0.2
0
10
20
30
40
50
60
70


model loss
train
test
loss
epoch
7000
6000
5000
4000
3000
2000
1000
0
0
10
20
30
40
50
60
70

4.2. Máy vector hỗ trợ - Support Vector Machine (SVM)

4.2.1. Tiền xử lý dữ liệu huấn luyện

- Việc tiền xử lí cũng được thực hiện giống như việc giải bài toán bằng CNN (phần 2a), các pixel có giá trị từ 0 đến 255 được đưa về phạm vi từ 0 đến 1 [0,1].
- Hình dưới cho thấy có 2 tập dữ liệu x (gồm 60000 elements) và x_test (gồm 10000 elements), mỗi tập dữ liệu gồm tập hợp các mảng 2 chiều kích thước (28x28) phạm vi từ [0,255] và được rút ngắn lại thành phạm vi [0,1]

```
x = x / 255.0
x_test = x_test / 255.0
```

4.2.2. Mô hình huấn luyện

- Radial Basic Function (RBF) kernel hay Gaussian kernel được sử dụng nhiều nhất trong thực tế, và là lựa chọn mặc định trong sklearn. Nó được định nghĩa bởi:

$$k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma||\mathbf{x} - \mathbf{z}||_2^2), \quad \gamma > 0$$

- Sigmoid kernel được sử dụng khá phổ biến trong thực tế. Nó được định nghĩa bởi:

$$f(s) = \frac{1}{1 + e^{-s}} \triangleq \sigma(s)$$

4.2.3. Kết quả sơ bộ

Score = 0.8829

	precision	recall	f1-score	support
0	0.83	0.86	0.84	1000
1	0.99	0.96	0.98	1000
2	0.79	0.82	0.80	1000
3	0.87	0.89	0.88	1000
4	0.81	0.81	0.81	1000
5	0.96	0.95	0.96	1000
6	0.72	0.66	0.69	1000
7	0.93	0.95	0.94	1000
8	0.97	0.98	0.97	1000
9	0.96	0.95	0.96	1000
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000

	precision	recall	f1-score	support
0	0.25	0.57	0.35	1000
1	0.88	0.65	0.75	1000
2	0.25	0.13	0.17	1000
3	0.65	0.41	0.50	1000
4	0.43	0.09	0.14	1000
5	0.61	0.58	0.59	1000
6	0.19	0.19	0.19	1000
7	0.90	0.64	0.75	1000
8	0.25	0.60	0.35	1000
9	0.90	0.47	0.61	1000
accuracy			0.43	10000
macro avg	0.53	0.43	0.44	10000
weighted avg	0.53	0.43	0.44	10000

Eslapse time : 2296.382757663727 seconds

	precision	recall	f1-score	support
0	0.78	0.83	0.80	1000
1	0.97	0.98	0.97	1000
2	0.75	0.82	0.78	1000
3	0.88	0.85	0.87	1000
4	0.81	0.77	0.79	1000
5	0.96	0.96	0.96	1000
6	0.70	0.65	0.68	1000
7	0.95	0.94	0.95	1000
8	0.96	0.96	0.96	1000
9	0.96	0.96	0.96	1000
accuracy			0.87	10000
macro avg	0.87	0.87	0.87	10000
weighted avg	0.87	0.87	0.87	10000

Eslapse time : 741.5575304031372 seconds

V. Đánh giá chung

5.1. Thuận lợi

- Các thành viên trong nhóm đều bắt tay làm việc khi đồ án được giao và hoàn thành đúng thời gian quy định.
- Công việc được phân chia rõ ràng cho từng thành viên, mỗi thành viên đều nhận thức rõ vai trò trách nhiệm của mình trong đồ án và có tinh thần tự giác cao.
- Các thành viên đều sẵn sàng hỗ trợ lẫn nhau khi gặp khó khăn hay vấn đề phát sinh trong lúc cài đặt tối ưu các thuật toán.
- Các thông tin về đồ án được giảng viên cung cấp đầy đủ giúp nhóm dễ dàng cài đặt các thuật toán, hoàn thành đồ án sớm nhất có thể.
- Giảng viên sẵn sàng giúp đỡ, giải đáp thắc mắc khi nhóm cần.

5.2. Khó khăn

- Do đây đều là những phương pháp nhận diện mới và tương đối phức tạp nên nhóm chưa từng có kinh nghiệm triển khai nên phải tốn nhiều thời gian để tìm hiểu.
- Bên cạnh đó, trong lúc xây dựng các mô hình nhận diện dựa vào những kiến thức vừa tìm hiểu nhóm gặp phải nhiều khó khăn như: thời gian làm đồ án tương đối ít để tìm hiểu sâu vấn đề , nhiều lỗi phát sinh trong lúc thực nghiệm. Tuy vậy, nhóm đã khắc phục được bằng việc trao đổi và hỗ trợ lẫn nhau để giải quyết những khó khăn phát sinh trong quá trình thực hiện đồ án.

5.3. Nhận xét chung về các mô hình

5.3.1. Mô hình mạng nơ-ron tích chập

Thuận lợi	Khó khăn
– Mô hình kiến trúc phức hợp giúp nâng cao độ chính xác – Với phần cứng TPU , thời gian huấn luyện được rút ngắn đáng kể – Mô hình có tính linh hoạt cao	– Mô hình phức tạp với nhiều phân lớp như Relu, BatchNormalization, Pooling và Dropout – Quá trình có xu hướng bị quá khớp (Overfit) – Cần có các phần cứng cao cấp như GPU hay TPU để huấn luyện – Dung lượng tệp lưu lại các tham số khá lớn

5.3.2. Mô hình SVM

Thuận lợi	Khó khăn
– Mô hình kiến trúc đơn giản – Mô hình có tính linh hoạt cao – Mô hình không cần có các phần cứng cao cấp để huấn luyện	– Quá trình có xu hướng bị quá khớp (Overfit) – Độ chính xác khá thấp so với mô hình còn lại – Thời gian huấn luyện khá lâu – Độ chính xác tùy thuộc kernel

VI. Tham khảo

[1] Z. C. L. M. L. A. J. S. Aston Zhang, "Pooling," [Online]. Available: http://d2l.ai/chapter_convolutional-neural-networks/pooling.html.

[2] T. H. Vu, "https://machinelearningcoban.com," 17 2 2017. [Online]. Available: https://machinelearningcoban.com/2017/02/17/softmax/#-cross-entropy.

[3] J. B. on, "A Gentle Introduction to Batch Normalization for Deep Neural Networks," 16 1 2020. [Online]. Available: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/.

[4] T. V. Huu, "Multi-layer Perceptron và Backpropagation," 24 2 2017. [Online]. Available: https://machinelearningcoban.com/2017/02/24/mlp/#-relu.

[5] A. Z. Karen Simonyan, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 10 4 2015. [Online]. Available: https://arxiv.org/abs/1409.1556.