

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**KHOA CÔNG NGHỆ THÔNG TIN I**

**A red circle with a yellow star in the middle

Description automatically generated with low confidence**

**BÁO CÁO**

**THỰC TẬP CƠ SỞ**

***Chủ đề: Nghiên cứu mô hình phân loại trạng phục Fashion-MNIST dựa trên học sâu.***

**Giảng viên : Đào Ngọc Phong**

**Họ và tên : Phạm Văn Thạch**

**Mã Sinh Viên : B20DCCN634**

**Số điện thoại : 0968966754**

# **LỜI CẢM ƠN**

Lời đầu tiên, em xin gửi lời cảm ơn đến Học viện Công nghệ Bưu chính Viễn Thông đã tạo điều kiện cho em được học môn Thực tập cơ sở. Đặc biệt, em xin gửi lời cảm ơn sâu sắc và chân thành nhất đến thầy Đào Ngọc Phong - người đã giúp đỡ em hoàn thành bài tập lớn môn thực tập cơ sở với chủ đề nghiên cứu mô hình phân loại trang phục Fashion-MNIST dựa trên học sâu.

Với vốn hiểu biết sâu rộng và kinh nghiệm nhiều năm giảng dạy cũng như làm việc trong môi trường công nghệ thông tin, thầy đã dành thời gian, kiến thức và tâm huyết để hướng dẫn và hỗ trợ em trong quá trình thực hiện bài tập. Nhờ sự chỉ dẫn và sự quan tâm tận tâm từ thầy, em đã có cơ hội nắm vững kiến thức và áp dụng thành công vào bài tập của mình. Sự kiên nhẫn, sự tận tụy và sự hỗ trợ của thầy đã giúp em vượt qua những khó khăn và trở thành một phiên bản tốt hơn của chính mình. Em rất biết ơn vì sự tận tâm và lòng nhiệt tình mà thầy dành cho em trong suốt quá trình làm bài.

Em xin cam kết sẽ nỗ lực hơn nữa và áp dụng những kiến thức và kỹ năng đã học được từ thầy vào công việc và hành trình phát triển của mình. Em sẽ luôn tự hào và biết ơn vì có một người thầy tuyệt vời như thầy.

Một lần nữa, em xin bày tỏ lòng biết ơn sâu sắc và chân thành đến thầy Đào Ngọc Phong. Rất mong có cơ hội được học hỏi và làm việc cùng thầy trong tương lai. Em xin chân thành cảm ơn và kính chúc Thầy luôn thành công và hạnh phúc trong sự nghiệp giảng dạy.

Trân trọng,

Thạch

Phạm Văn Thạch

# 

# **I. Giới thiệu về tập dữ liệu FASHION-MNIST**

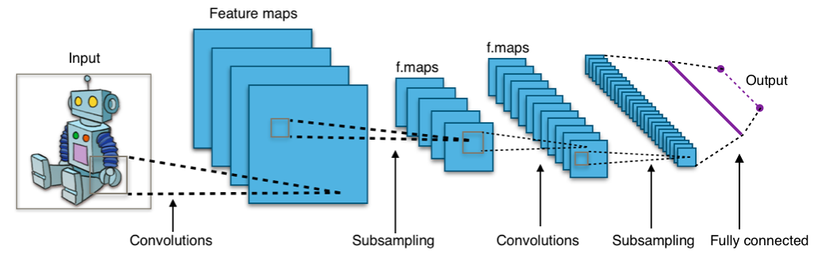
* Ngành công nghiệp thời trang ngày càng phát triển mạnh mẽ, đáp ứng nhu cầu ngày càng cao của khách hàng. Sự phát triển của công nghệ và cuộc cách mạng số đã tạo ra sự gia tăng về khả năng thu thập và xử lý dữ liệu hình ảnh, đặc biệt là trong lĩnh vực nhận dạng và phân loại sản phẩm thời trang.
* Trong quá trình mua sắm trực tuyến hoặc tại cửa hàng, việc nhận diện và phân loại sản phẩm thời trang đúng, nhanh chóng và chính xác là rất quan trọng. Điều này giúp cải thiện trải nghiệm mua sắm của người dùng, giảm thời gian tìm kiếm và tăng khả năng tìm kiếm các sản phẩm tương tự.
* Tập dữ liệu Fashion-MNIST đã được giới thiệu nhằm đáp ứng nhu cầu nghiên cứu và phát triển các mô hình học máy trong lĩnh vực nhận dạng và phân loại sản phẩm thời trang. Tập dữ liệu này được xây dựng dựa trên ý tưởng của tập dữ liệu MNIST, một tập dữ liệu rất phổ biến trong lĩnh vực nhận dạng chữ số viết tay.
* Fashion-MNIST là một tập dữ liệu về 70.000 hình ảnh, trong đó 60.000 ảnh sẽ được dùng để huấn luyện và 10.000 ảnh sẽ thường dùng để đánh giá khả năng phân loại nhận diện ảnh của mạng neuron. Mỗi ảnh là một mảng NumPy 2 chiều, 28x28, với mỗi pixel có giá trị từ 0 đến 255.

\*Lý do chọn Fashion-MNIST:

* Độ phức tạp tương đối: So với tập dữ liệu MNIST gốc chỉ chứa các hình ảnh chữ số viết tay, Fashion-MNIST cung cấp một thách thức phức tạp hơn với việc nhận diện và phân loại các đồng phục và sản phẩm thời trang. Điều này phản ánh chính xác hơn các tình huống thực tế trong ngành thời trang.
* Đa dạng và phong phú: Tập dữ liệu Fashion-MNIST bao gồm 10 lớp đồng phục và sản phẩm thời trang khác nhau, từ áo phông, váy, áo khoác đến giày dép và túi xách. Đây là một sự đa dạng đáng kể, giúp mô hình học máy có khả năng phân loại và nhận dạng nhiều loại sản phẩm.
* Dung lượng nhỏ gọn: Tập dữ liệu Fashion-MNIST có kích thước nhỏ gọn so với một số tập dữ liệu hình ảnh lớn khác. Với tổng cộng 70.000 hình ảnh, tập dữ liệu này dễ dàng tiếp cận và xử lý trên các tài nguyên hạn chế.
* Độ khó điều chỉnh: Mặc dù Fashion-MNIST không phức tạp như các tập dữ liệu thời trang thực tế, nó vẫn cung cấp độ khó điều chỉnh phù hợp để nghiên cứu và đánh giá các mô hình học máy. Nó cung cấp một bước đi đầu tiên để nắm bắt và hiểu quy trình phân loại và nhận dạng sản phẩm thời trang.
* Tiềm năng ứng dụng rộng rãi: Kết quả nghiên cứu và phát triển trên tập dữ liệu Fashion-MNIST có thể được ứng dụng trong nhiều lĩnh vực, từ công nghệ thời trang, gợi ý mua sắm, cho đến phân loại sản phẩm và phân tích xu hướng thị trường.

# **II. Giới thiệu về kiến trúc CNN**

* Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một tập hợp các lớp Convolutional chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Đây một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.



* CNN bao gồm những phần lớp cơ bản là:

## **Convolutional layer**

Convolutional layer (lớp tích chập) là một thành phần quan trọng trong mạng nơ-ron tích chập (CNN). Lớp này thực hiện phép tích chập trên dữ liệu đầu vào để trích xuất các đặc trưng cục bộ. Nếu lớp này có số lượng lớn có thể dẫn đến tác động được giảm một cách đáng kể.

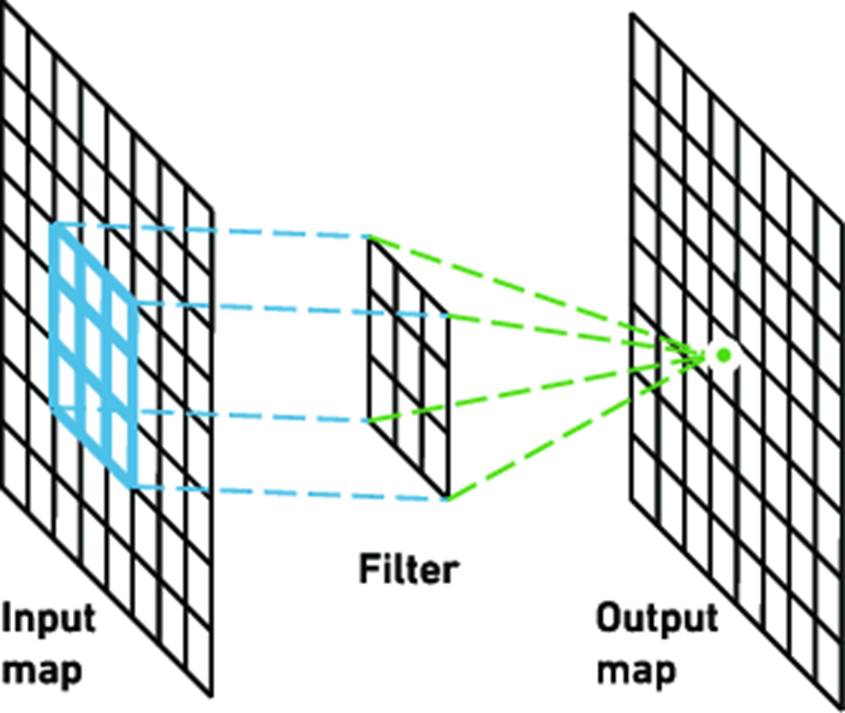
Thường thì chỉ sau 3 đến 4 layer thôi là ta sẽ đạt được kết quả như mong muốn. Những yếu tố quan trọng của một convolutional layer là: stride, padding, filter map, feature map.

CNN sử dụng các filter map để áp dụng vào vùng của hình ảnh. Những filter map này được gọi là ma trận 3 chiều, mà bên trong nó là các con số và chúng là parameter.

+Stride có nghĩa là khi bạn dịch chuyển filter map theo pixel dựa vào giá trị trừ trái sang phải. Và sự chuyển dịch này chính là Stride.

+Padding: Là các giá trị 0 được thêm vào với lớp input.

+Feature map: Nó thể hiện kết quả của mỗi lần filter map quét qua input. Sau mỗi lần quét sẽ xảy ra quá trình tính toán.



*\*Một số tính chất quan trọng của lớp tích chập là:*

+ Các bộ lọc được chia sẻ: Cùng một bộ lọc được sử dụng trên toàn bộ dữ liệu đầu vào, cho phép chia sẻ trọng số và giảm số lượng tham số trong mô hình.

+ Tích chập với các bước (strides): Bước là khoảng cách giữa các vị trí bắt đầu của bộ lọc khi trượt trên dữ liệu. Tích chập với bước nhỏ hơn 1 sẽ tạo ra các feature map có kích thước nhỏ hơn đầu vào.

+ Độ sâu (depth) của feature map: Mỗi lớp tích chập sẽ áp dụng nhiều bộ lọc khác nhau để tạo ra nhiều feature map có độ sâu khác nhau. Điều này cho phép mô hình học được các đặc trưng ở mức độ trừu tượng và phức tạp hơn.

## **Pooling layer**

Lớp chứa hay lớp tổng hợp (Pooling layer): Là những lớp có tác dụng làm đơn giản hóa các thông tin ở đầu ra từ các lớp tích chập, giúp làm giảm parameter khi đầu vào quá lớn. Lớp pooling thường được sử dụng ngay sau lớp tích chập.

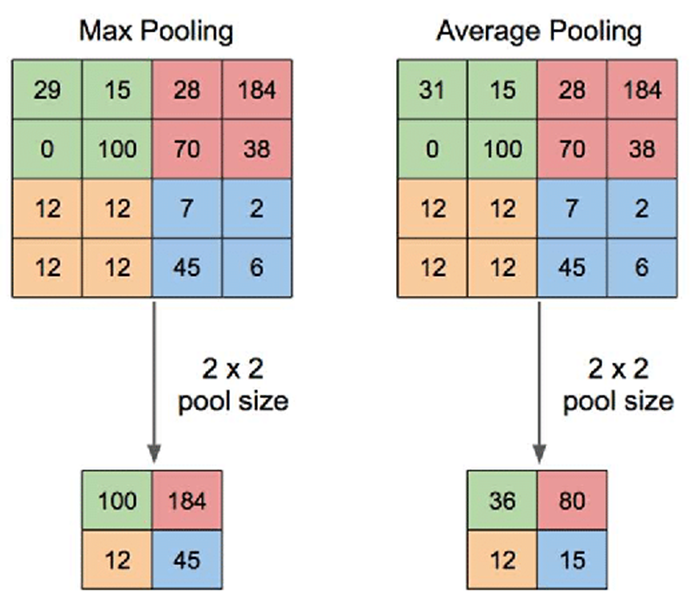
Có 2 loại pooling layer phổ biến là: max pooling và average pooling:

1. Max Pooling:

* Phép Max Pooling lấy giá trị lớn nhất từ mỗi vùng con của đầu vào. Vùng con được xác định bằng cách chống lên đầu vào và di chuyển theo bước(stride). Max Pooling giữ lại đặc trưng quan trọng nhất từ mỗi vùng con và giảm kích thước không gian của đặc trưng. Điều này giúp giảm overfitting và tăng tính tổng quát hóa của mô hình.

1. Average Pooling:

* Phép average Pooling lấy giá trị trung bình từ mỗi vùng con của đầu vào. Tương tự như Max Pooling, Average Pooling cũng giảm kích thước không gian của đặc trưng, nhưng nó có xu hướng mượt hơn và giảm các đặc trưng nhiễu.



*Nguyên lý hoạt động của 2 loại pooling*

Mục đích của pooling rất đơn giản, nó làm giảm số siêu tham số mà ta cần phải tính toán, từ đó giảm thời gian tính toán, tránh overfitting. Loại pooling ta thường gặp nhất là max pooling, lấy giá trị lớn nhất trong một cửa sổ pooling. Pooling hoạt động gần giống với tích chập, nó cũng có 1 cửa sổ trượt gọi là pooling window, cửa sổ này trượt qua từng giá trị của ma trận dữ liệu đầu vào, chọn ra một giá trị từ các giá trị nằm trong cửa sổ trượt (với max pooling ta sẽ lấy giá trị lớn nhất). Ví dụ như hình vẽ trên ta chọn pooling window có kích thước là 2 \* 2, stride = 2 để đảm bảo không trùng nhau, và áp dụng max pooling hoặc average pooling.

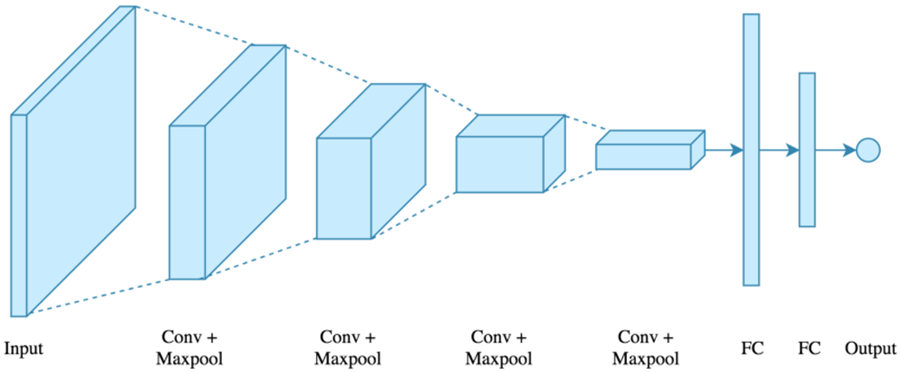
## **Fully connected layer**

Lớp này có nhiệm vụ đưa ra kết quả sau khi lớp convolutional layer và pooling layer đã nhận được ảnh truyền. Lúc này, ta thu được kết quả là model đã đọc được thông tin của ảnh và để liên kết chúng cũng như cho ra nhiều output hơn thì ta sử dụng fully connected layer.

Ngoài ra, nếu như fully connected layer có được dữ liệu hình ảnh thì chúng sẽ chuyển nó thành mục chưa được phân chia chất lượng. Cái này khá giống với phiếu bầu rồi chúng sẽ đánh giá để bầu chọn ra hình ảnh có chất lượng cao nhất.

Giống trong mạng nơ ron thì mỗi lớp ẩn được gọi là kết nối đầy đủ - fully connected. Thường sau lớp kết nối đầy đủ sẽ là 2 lớp kết nối đầy đủ, 1 lớp để tập hợp các lớp đặc trưng mà ta đã tìm ra, chuyển đổi dữ liệu từ 3D, hoặc 2D thành 1D, tức chỉ còn là 1 vector. Còn 1 lớp nữa là đầu ra, số nơ ron của lớp này phụ thuộc vào số đầu ra mà ta muốn tìm ra.

\***Ví dụ một cấu trúc mạng CNN tiêu chuẩn**



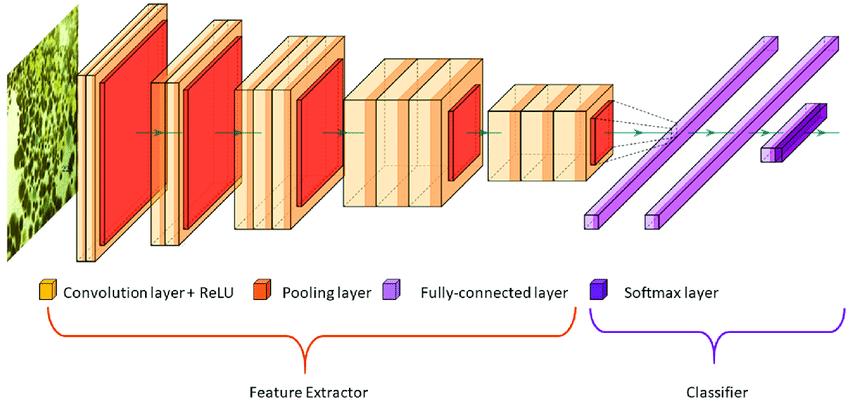
Trên tư tưởng của một mạng CNN tiêu chuẩn cùng với sự đóng góp dữ liệu từ thử thách nhận dạng hình ảnh quy mô lớn ImageNet hoặc ILSVRC đã đem đến nhiều mô hình CNN có độ chính xác ngày được cải thiện và nhiều hơn các biến thể CNN được ra đời. Một số các mô hình CNN, biến thể CNN nổi bật có những kết quả cao trong bài toán phân loại ảnh.

Để đánh giá một mô hình thì trước hết cần có một tập dữ liệu tiêu chuẩn, một thước đo cho sự chính xác đảm bảo tính công bằng. Trong bài toán phân loại ảnh thì ImageNet được xem là một trong tập dữ liệu đáng tin cậy và trực quan nhất đánh giá các mô hình nhận dạng ảnh thông qua cuộc thi hằng năm.

# **III. Giới thiệu về kiến trúc VGG-16.**

1. **Giới thiệu.**

**-** VGG16 là một kiến trúc mạng neural convolutional (CNN) được xây dựng trên cơ sở các lớp convolutional có kích thước nhỏ và sâu, với các lớp convolutional được xếp chồng lên nhau liên tiếp nhau. Kiến trúc này bao gồm 13 lớp convolutional và 3 lớp fully connected, tổng cộng 16 lớp.



- Các lớp convolutional đầu tiên của VGG16 sử dụng các bộ lọc có kích thước 3x3 và stride = 1, được áp dụng trên đầu vào ảnh. Sau đó, các lớp max pooling với kích thước 2x2 được sử dụng để giảm kích thước của đầu ra từ các lớp convolutional trước đó.

- Sau đó, các lớp convolutional tiếp theo tiếp tục sử dụng các bộ lọc 3x3 với stride = 1. Tuy nhiên, các lớp convolutional này được thiết kế với số lượng bộ lọc lớn hơn và độ sâu cao hơn so với các lớp convolutional trước đó.

- Cuối cùng, các lớp fully connected được sử dụng để phân loại ảnh vào các lớp nhãn khác nhau. VGG16 sử dụng một lớp fully connected với 4096 nơ-ron và một lớp fully connected cuối cùng với số lượng nơ-ron bằng với số lượng lớp nhãn trong bộ dữ liệu huấn luyện.

1. **Ưu, nhược điểm**

*\* Ưu điểm của VGG16:*

* VGG16 có độ chính xác cao trên các tập dữ liệu phân loại ảnh nổi tiếng như ImageNet.
* Kiến trúc đơn giản và dễ hiểu, cho phép nghiên cứu và phát triển dễ dàng.
* Có thể tái sử dụng và chuyển giao cho các bài toán khác nhau.

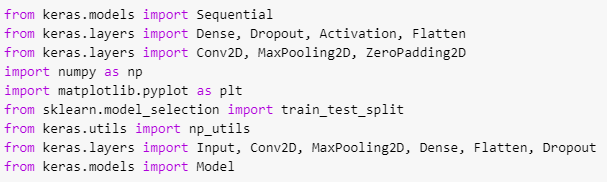
*\*Nhược điểm của VGG16:*

* Kiến trúc phức tạp hơn so với một số mô hình CNN khác, dẫn đến thời gian huấn luyện lâu hơn.
* Có thể gặp vấn đề về overfitting trên các tập dữ liệu nhỏ hoặc không đủ đa dạng.
* Số lượng tham số lớn, dẫn đến khó khăn trong việc triển khai trên các thiết bị có tài nguyên hạn chế.

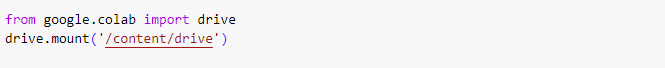
Trong những năm gần đây, các kiến trúc mạng nơ-ron sâu khác như ResNet, Inception và EfficientNet đã được phát triển và được sử dụng rộng rãi hơn. Các kiến trúc này giải quyết được những nhược điểm của VGG-16 và đạt được độ chính xác cao hơn trong nhiều bài toán nhận dạng hình ảnh.

# **IV. Các bước giải bài toán.**

## **Khai báo thư viện.**



* Kết nối Drive với Google Colab



## **Tiền xử lý dữ liệu**

### **2.1. Tải xuống và đọc tập dữ liệu.**

* Import tập dữ liệu Fashion MNIST trong thư viện Keras

from keras.datasets import fashion\_mnist

* Tách Fashion-Mnist thành 2 tập train và test:

(x\_train, y\_train), (x\_test, y\_test) =fashion\_mnist.load\_data()

* Chuẩn hóa hình ảnh:

+ Trong tập dữ liệu Fashion MNIST, các giá trị pixel của các hình ảnh nằm trong khoảng từ 0 đến 255. Đây là giá trị của mức xám trong đó 0 đại diện cho đen tuyệt đối và 255 đại diện cho màu trắng tuyệt đối. Khi sử dụng hình ảnh như vậy và chuyển qua mạng nơron sâu, việc tính toán các giá trị số cao có thể trở nên phức tạp hơn.

+ Trước khi sử dụng tập dữ liệu Fashion MNIST trong quá trình huấn luyện mô hình, ta thường tiến hành chuẩn hóa các giá trị pixel để nằm trong khoảng từ 0 đến 1. Điều này có thể được thực hiện bằng cách chia các giá trị pixel cho 255 để đưa chúng về khoảng [0,1]. Quá trình chuẩn hóa này giúp việc tính toán trở nên dễ dàng và nhanh chóng hơn cải thiện khả năng học và tối ưu hóa mô hình.

x\_train = x\_train/255

x\_test = x\_test/255

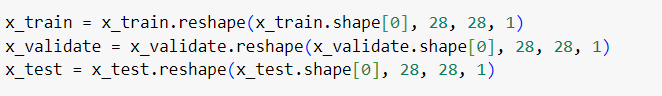
**2.2: Tách dữ liệu để đánh giá hiệu suất của mô hình.**

* Tách tiếp tập train thành x\_train, y\_train (80% dữ liệu) sẽ được sử dụng để huấn luyện mô hình, và x\_validate, y\_validate (20% dữ liệu) sẽ được sử dụng để đánh giá hiệu suất của mô hình. random\_state = 0 được sử dụng để đảm bảo rằng mỗi lần chạy mã sẽ cho kết quả chia dữ liệu tương tự nhau.

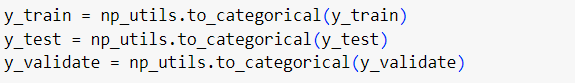
x\_train, x\_validate, y\_train, y\_validate = train\_test\_split(x\_train, y\_train, test\_size=0.2, random\_state=0)

**2.3: Reshape dữ liệu, mã hóa one-hot.**

* Reshape dữ liệu là quan trọng để phù hợp với đầu vào của mô hình. Kích thước 28x28x1 tương ứng với kích thước của một hình ảnh xám có chiều sâu 1. Điều này cần thiết để có thể áp dụng các lớp tích chập và các phép tính trên hình ảnh trong mạng nơ-ron tích chập.



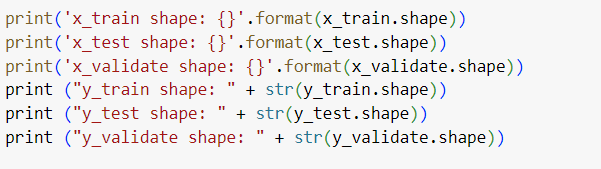
* Mã hóa one-hot: là biểu diễn của các biến phân loại dưới dạng vectơ nhị phân. Điều này đầu tiên yêu cầu các giá trị phân loại được ánh xạ thành các giá trị số nguyên. Sau đó, mỗi giá trị số nguyên được biểu diễn dưới dạng vectơ nhị phân có tất cả các giá trị bằng 0 ngoại trừ chỉ số của số nguyên, được đánh dấu bằng 1.
* One-hot encoding là một quá trình mà các biến phân loại (label) được chuyển đổi thành một mẫu có thể cung cấp cho các thuật toán ML để thực hiện công việc tốt hơn khi mà dự đoán.



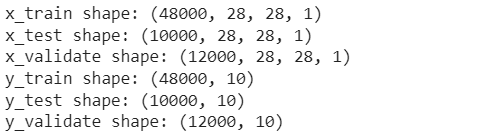
Ví dụ: Ta có dữ liệu 1 cột là: “Áo thun”, “Quần dài”,”Túi xách”. Xây dựng từ điển [“Áo thun”, “Quần dài”,”Túi xách”], tiếp theo cần lưu lại chỉ số của cùng hạng mục trong từ điển: “Áo thun”: 0, ”Quần dài”: 1, “Túi xách”: 2

|  |  |
| --- | --- |
| Giá trị | Mã one-hot |
| “Áo thun” | [1, 0, 0] |
| “Quần dài” | [0, 1, 0] |
| “Túi xách” | [0, 0, 1] |

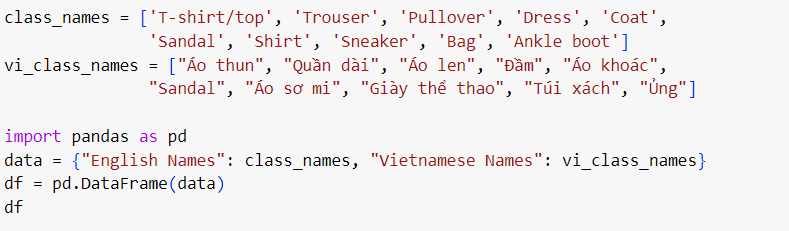
* Kiểm tra và in ra kích thước (shape) của các biến:



=> Kết quả:

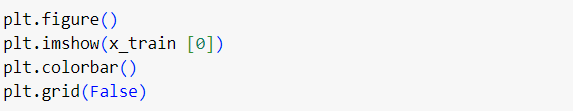


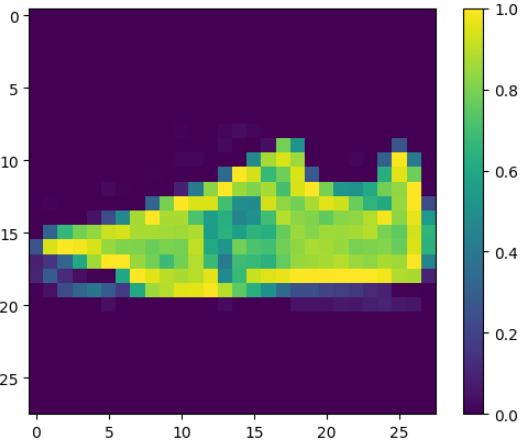
**2.4: Tạo 1 bảng dữ liệu để hiển thị thông tin về tên lớp trong bộ dữ liệu Fashion MNIST và tên tiếng Việt tương ứng.**



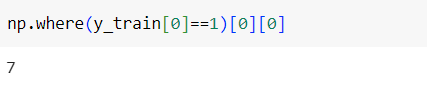


* Hiển thị hình ảnh từ tập dữ liệu x\_train (hình ảnh đầu tiên)



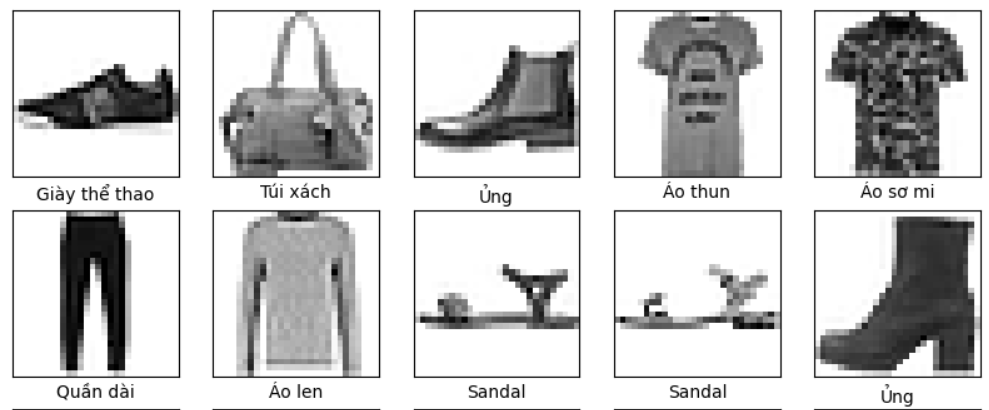


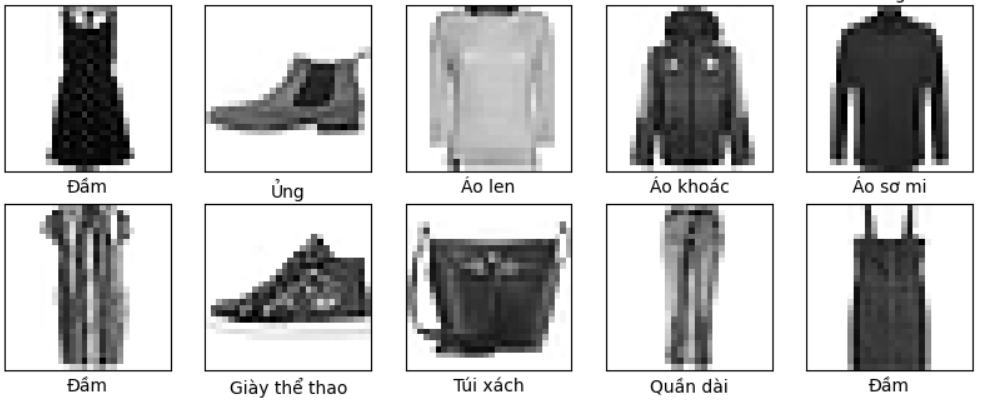
=> Hiển thị nhãn tương ứng với hình ảnh từ tập y\_train:



=> Nhãn ứng với hình ảnh trên là 7 => Sneaker (Giày thể thao)

* Hiển thị 20 hình ảnh đầu của tập train và nhãn tương ứng:





## **Xây dựng mạng CNN**

### **3.1. Một số khái niệm**

#### **3.1.1.Keras:**

Keras là một open source cho Neural Network được viết bởi ngôn ngữ Python. Nó là một library được phát triển vào năm 205 bởi Francois Chollet, là một kỹ sư nghiên cứu Deep Learning. Keras có thể sử dụng chung với các thư viện nổi tiếng như Tensorflow, CNTK, Theano. Một số ưu điểm của Keras như:

* Dễ sử dụng, dùng đơn giản hơn Tensor, xây dựng model nhanh.
* Run được trên cả CPU và GPU.
* Hỗ trợ xây dựng CNN, RNN hoặc cả hai.

#### **3.1.2. Sequential model:**

- Sequential model là một kiểu mô hình trong thư viện Keras cho phép xây dựng mạng nơ-ron tuần tự (sequential neural network) dễ dàng và linh hoạt. Trong kiểu mô hình này, các layer (lớp) được xếp chồng lên nhau theo một thứ tự tuần tự, từ đầu vào đến đầu ra.

- Mạng Sequential model trong Keras cho phép ta xây dựng một chuỗi các lớp (layer) tuần tự một cách dễ dàng. Mỗi lớp trong mạng Sequential model đại diện cho một phần tử của mạng nơ-ron, ví dụ như lớp convolutional, lớp pooling, lớp fully connected, lớp activation và nhiều lớp khác.

- Một ưu điểm quan trọng của Sequential model là sự đơn giản và dễ sử dụng. Với việc xếp chồng các lớp theo thứ tự tuần tự, ta có thể dễ dàng xây dựng và tùy chỉnh mạng nơ-ron một cách linh hoạt. Ta chỉ cần thêm các lớp mong muốn vào mạng theo đúng thứ tự từ đầu đến cuối.

#### **3.1.3. Tạo Convolutional layer:**

Conv2D là convolution dùng để lấy feature từ ảnh với các tham số :

* filters : số filter của convolution
* kernel\_size : kích thước window search trên ảnh
* strides : số bước nhảy trên ảnh
* activation : chọn activation như linear, softmax, relu, tanh, sigmoid.
* padding : có thể là "valid" hoặc "same". Với same thì có nghĩa là padding =1.

**3.1.4: Hàm Relu:**

Hàm ReLU (Rectified Linear Unit) là một hàm kích hoạt phổ biến trong thuật toán Mạng nơ-ron tích chập (Convolutional Neural Network - CNN). Hàm này được sử dụng để đưa ra quyết định về việc kích hoạt (hoạt động) hoặc tắt (vô hoạt động) các nơ-ron trong mô hình CNN.

Hàm ReLU được định nghĩa đơn giản: nếu đầu vào là dương, nó sẽ trả về chính giá trị đó; nếu đầu vào là âm, nó sẽ trả về 0. Công thức toán học của hàm ReLU có thể được biểu diễn như sau:

ReLU(x) = max(0, x)

Trong đó:

- x là đầu vào của hàm ReLU.

- max(a, b) là một hàm trả về giá trị lớn nhất giữa hai số a và b.

Hàm ReLU có những đặc điểm quan trọng giúp nó trở thành lựa chọn phổ biến trong CNN:

*1. Phi tuyến tính:* Hàm ReLU có tính chất phi tuyến tính, giúp mô hình CNN có khả năng học được các mô hình phức tạp và khái quát hóa tốt hơn.

*2. Không bị biến mất đạo hàm:* Một vấn đề phổ biến trong các hàm kích hoạt khác như Sigmoid hay Tanh là hiện tượng biến mất đạo hàm (vanishing gradient) khi độ sâu của mạng tăng lên. Hàm ReLU không gặp phải vấn đề này vì đạo hàm của nó luôn bằng 1 với đầu vào dương.

*3. Hiệu suất tính toán:* Hàm ReLU có tính chất đơn giản và tính toán hiệu quả. So với các hàm kích hoạt khác, việc tính toán hàm ReLU nhanh chóng và không tốn nhiều tài nguyên tính toán.

Hàm ReLU thường được sử dụng sau các phép tích chập và trước các lớp kết nối đầy đủ (fully connected layers) trong mạng CNN. Nó giúp tạo ra các đặc trưng phi tuyến tính và tăng tính khái quát hóa của mô hình.

#### **3.1.5. Pooling layer:**

Được sử dụng để làm giảm param khi train, nhưng vẫn giữ được đặc trưng của ảnh.

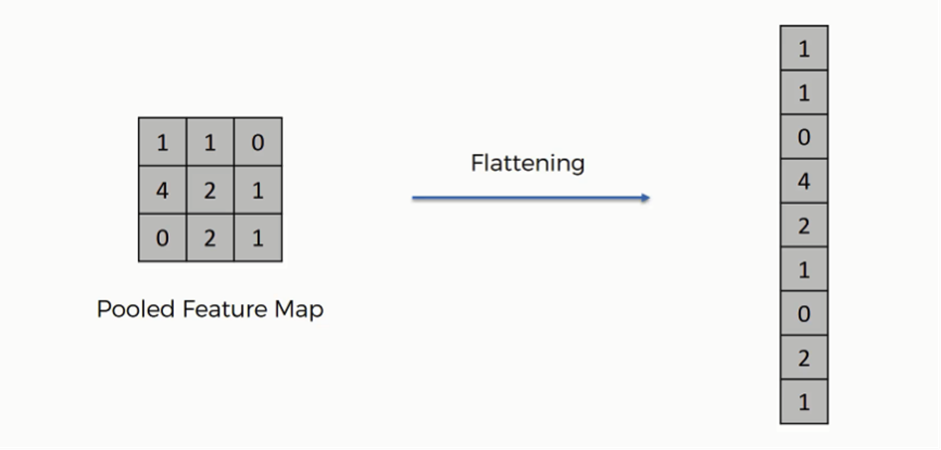
* pool\_size : kích thước ma trận để lấy max hay average
* Ngoài ra còn có : MaxPooling2D, AvergaPooling1D, 2D ( lấy max , trung bình) với từng size.

Pooling layer cũng có vai trò giúp mạng nơ-ron trở nên không nhạy với việc tịnh tiến nhỏ, tức là dữ liệu đầu vào có thể được dịch chuyển một chút mà không ảnh hưởng đáng kể đến kết quả. Điều này có ý nghĩa trong việc nhân diện đối tượng trong hình ảnh, nơi vị trí của đối tượng có thể thay đổi.

**3.1.6. Flatten:**

Bước làm phẳng là một bước rất đơn giản liên quan đến việc xây dựng một mạng nơ-ron tích hợp. Nó liên quan đến việc lấy pooled feature map được tạo ra trong bước gộp và biến nó thành vectơ một chiều.

Lý do tại sao chúng ta biến đổi pooled feature map thành vectơ một chiều là vì vectơ này bây giờ sẽ được đưa vào một [mạng nơ-ron nhân tạo](https://nickmccullum.com/python-deep-learning/artificial-neural-network-tutorial/). Nói cách khác, vectơ này bây giờ sẽ trở thành lớp đầu vào của mạng nơ-ron nhân tạo sẽ được xâu chuỗi vào mạng nơ-ron tích tụ mà chúng tôi đã xây dựng cho đến nay trong khóa học này.

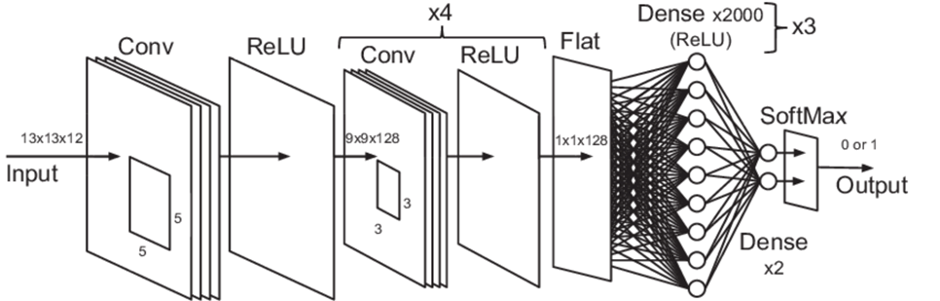


**3.1.7. Dense:**

- Lớp Dense trong mạng neural được sử dụng để tạo ra các kết nối đầy đủ giữa các nơ-ron trong các lớp liền kề. Nó là một lớp hồi quy (regression) hoặc lớp phân loại (classification) thông thường trong các mô hình mạng neural.

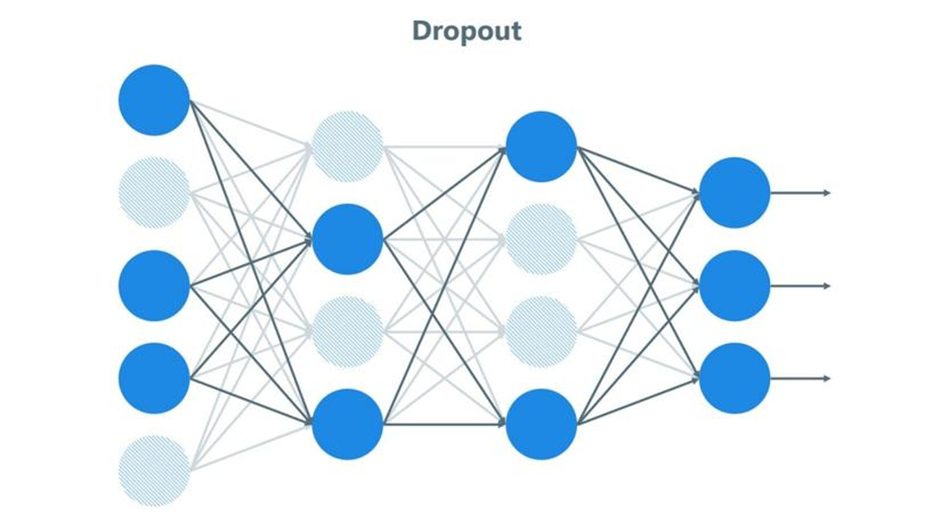
- Lớp Dense thường được sử dụng làm lớp cuối cùng của một mô hình mạng neural, để biểu diễn kết quả cuối cùng của mô hình. Ví dụ, trong bài toán phân loại, lớp Dense cuối cùng có thể sử dụng hàm kích hoạt softmax để tính toán xác suất của các lớp phân loại khác nhau.

- Thông qua việc sử dụng các lớp Dense, mô hình mạng neural có thể học được các đặc trưng phức tạp và tạo ra các dự đoán chính xác trong các bài toán hồi quy và phân loại.



#### **3.1.8. Dropout:**

Trong mạng neural network, kỹ thuật dropout là việc chúng ta sẽ bỏ qua một vài unit trong suốt quá trình train trong mô hình, những unit bị bỏ qua được lựa chọn ngẫu nhiên. Ở đây, chúng ta hiểu “bỏ qua - ignoring” là unit đó sẽ không tham gia và đóng góp vào quá trình huấn luyện (lan truyền tiến và lan truyền ngược).



**3.1.9. Hàm compile:**

Trong xây dựng mạng CNN, Hàm compile được sử dụng để cấu hình mô hình trước khi thực hiện quá trình huấn luyện. Hàm compile thường được gọi sau khi mô hình đã được xác định và các layer đã được thêm vào.

Hàm compile có vai trò quyết định các thông số quan trọng của quá trình huấn luyện, bao gồm hàm mất mát (loss function), thuật toán tối ưu (optimizer) và các metrics (đánh giá hiệu suất). Dưới đây là những khái niệm cơ bản liên quan đến các thông số này:

**3.1.9.1. Loss function:**

* Hàm mất mát là một hàm được sử dụng để đo lường mức độ sai lệch giữa giá trị dự đoán và giá trị thực tế. Ví dụ, trong bài toán phân loại, hàm mất mát thường là categorical cross-entropy hoặc binary cross-entropy. Mục tiêu là tìm cách tối thiểu hóa giá trị mất mát này trong quá trình huấn luyện.

**3.1.9.2. Optimizer:**

* Thuật toán tối ưu là một phương pháp được sử dụng để tìm kiếm bộ trọng số tối ưu cho mạng nơ-ron. Ví dụ, các thuật toán phổ biến như Stochastic Gradient Descent (SGD), Adam, RMSprop được sử dụng để điều chỉnh và cập nhật trọng số trong quá trình huấn luyện.

**3.1.9.3. Metrics:**

* Metrics là các độ đo được sử dụng để đánh giá hiệu suất của mô hình. Ví dụ, trong bài toán phân loại, metrics phổ biến là accuracy (độ chính xác), precision (độ chính xác dương tính), recall (độ bao phủ) và F1-score (độ đo kết hợp giữa precision và recall). Metrics giúp ta đánh giá và theo dõi hiệu suất mô hình trong quá trình huấn luyện và kiểm tra.

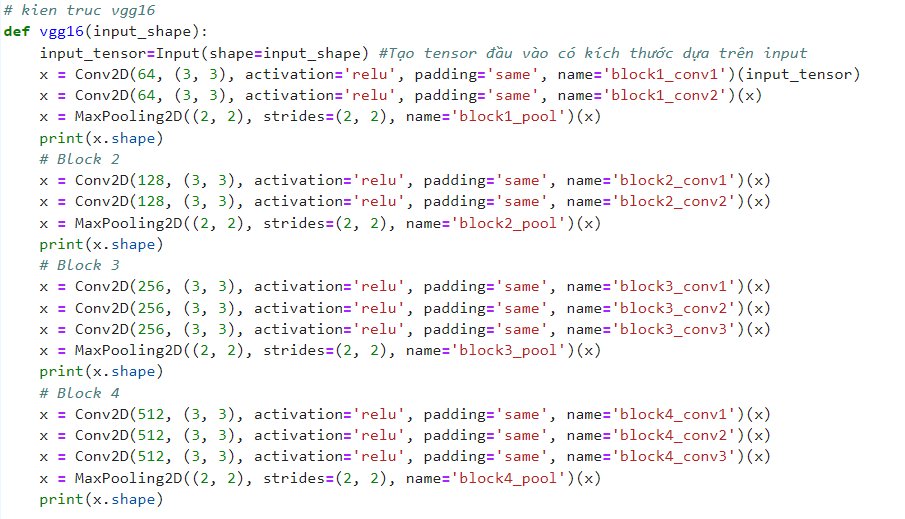
=> Khi gọi hàm compile, ta cần chỉ định các thông số trên, bằng cách truyền vào các đối số tương ứng. Hàm compile sẽ tiến hành cấu hình mô hình dựa trên các thông số đã được xác định, chuẩn bị cho quá trình huấn luyện.

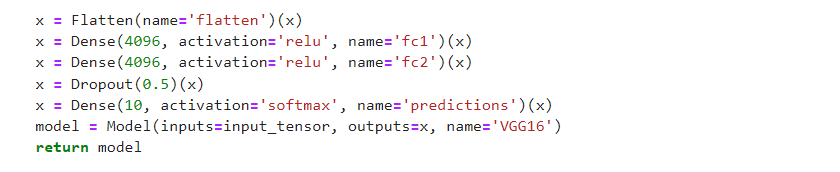
#### **3.1.10. Hàm fit:**

Hàm fit() trong mô hình mạng neural là một phương thức được sử dụng để huấn luyện mô hình trên dữ liệu đào tạo. Hàm này thực hiện quá trình tối ưu hóa các tham số của mạng neural để cung cấp các dự đoán chính xác cho dữ liệu mới.

* Bao gồm data train, test đưa vào training.
* Batch\_size thể hiện số lượng mẫu mà Mini-batch GD sử dụng cho mỗi lần cập nhật trọng số .
* Epoch là số lần duyệt qua hết số lượng mẫu trong tập huấn luyện.

### **3.2. Xây dựng mô hình**





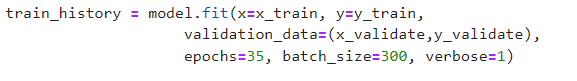
### **3.3. Quy trình đào tạo**

#### **3.3.1: Xác định quá trình đào tạo, xác định tham số:**

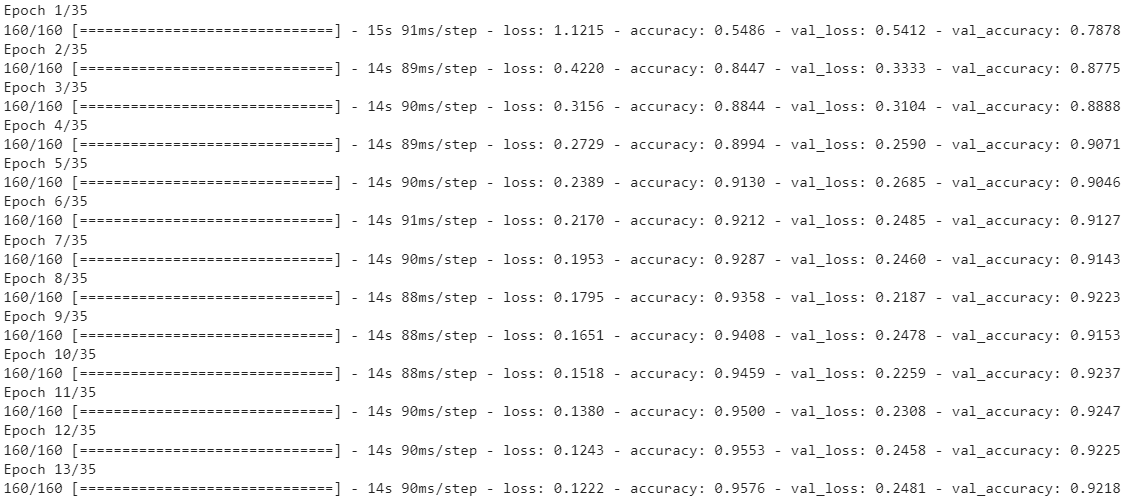


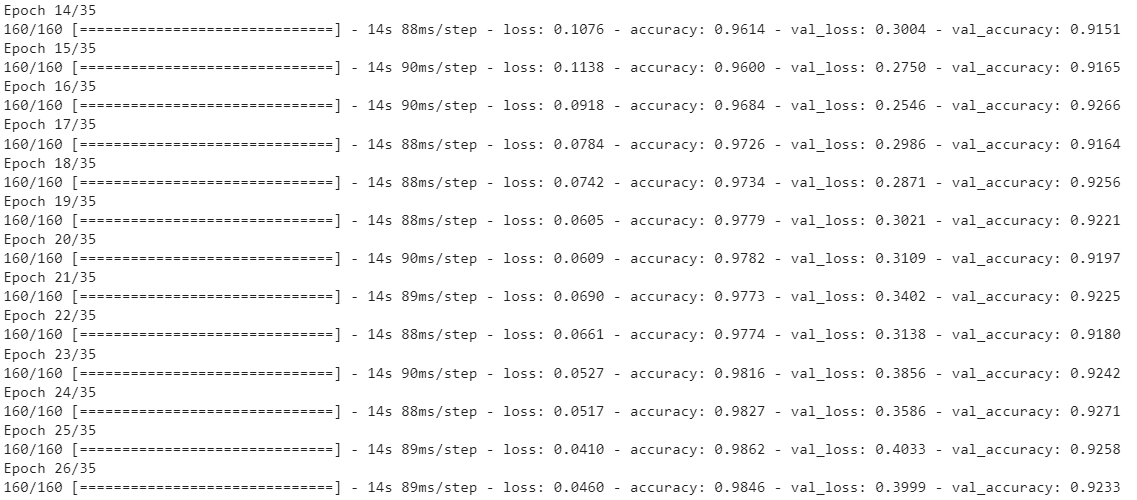
* Thuật toán tối ưu (optimizer): Bộ tối ưu hóa Adam là một thuật toán tối ưu hóa thích hợp cho việc huấn luyện mạng nơ-ron
* Hàm mất mát (loss function): được sử dụng để đo lường hiệu suất của mô hình trong quá trình huấn luyện. Hàm mất mát categorical\_crossentropy phù hợp với bài toán phân loại đa lớp.
* Metrics: Tham số đánh giá hiệu suất mô hình trong quá trình huấn luyện và kiểm tra. Accuracy là độ đo phổ biến để đánh giá hiệu suất của mô hình phân loại, nó cho biết tỷ lệ dự đoán chính xác trên tổng số dữ liệu. Mô hình sẽ tính toán và hiển thị giá trị độ chính xác sau mỗi vòng lặp trong quá trình huấn luyện (epochs).

#### **3.3.2: Quá trình huấn luyện mô hình sử dụng dữ liệu huấn luyện và dữ liệu validation.**



* model.fit(): Phương thức dùng để huấn luyện mô hình, nó nhận vào các đối số như dữ liệu huấn luyện, nhãn, dữ liệu validation, số epochs, kích thước batch, và verbose
* epochs: là số lượng vòng lặp huấn luyện, mỗi epoch tương ứng với một lần lặp toàn bộ dữ liệu huấn luyện.
* batch\_size: là kích thước của từng batch dữ liệu sử dụng trong quá trình huấn luyện, mỗi batch gồm 300 mẫu được đưa vào mô hình để tính toán đạo hàm và cập nhật trọng số.
* verbose: Tham số xác định cách hiển thị thông tin trong quá trình huấn luyện. Giá trị ‘1’ được sử dụng để hiển thị tiến trình huấn luyện bao gồm thông tin về số epoch, loss, accuracy trên dữ liệu huấn luyện và validation.

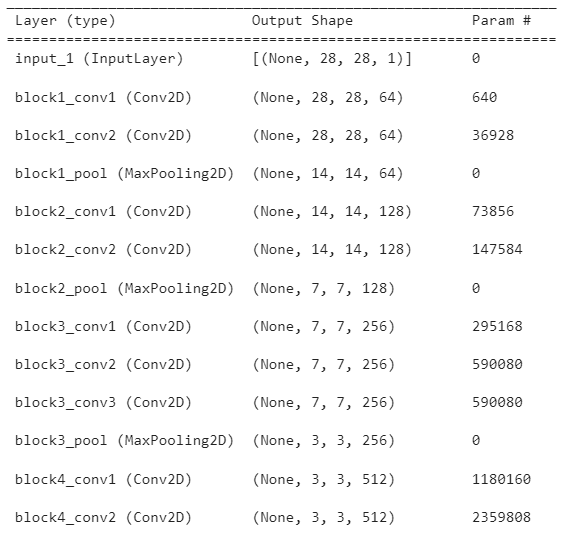


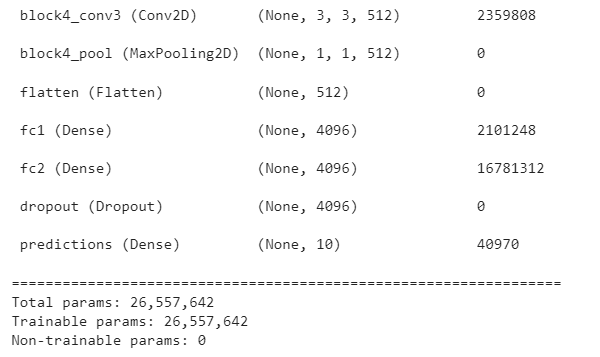




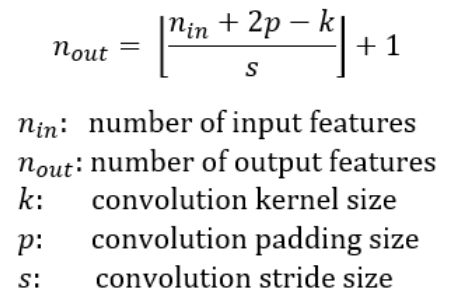
=> Qua quá trình huấn luyện: giá trị đo lường mức độ sai khác giữa giá trị dự đoán của mô hình và giá trị thực tế trong quá trình huấn luyện (loss) đang giảm dần qua từng vòng lặp huấn luyện (epoch), còn độ chính xác (accuracy) càng tăng cao chứng tỏ mô hình đang có khả năng dự đoán chính xác hơn trên dữ liệu huấn luyện

#### **3.3.3: Hiển thị thông tin về lịch sử huấn luyện**

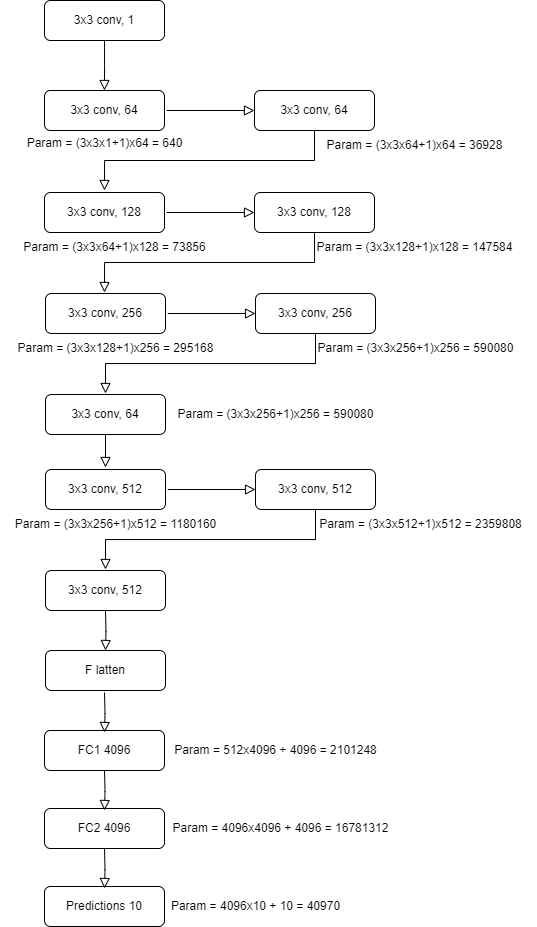




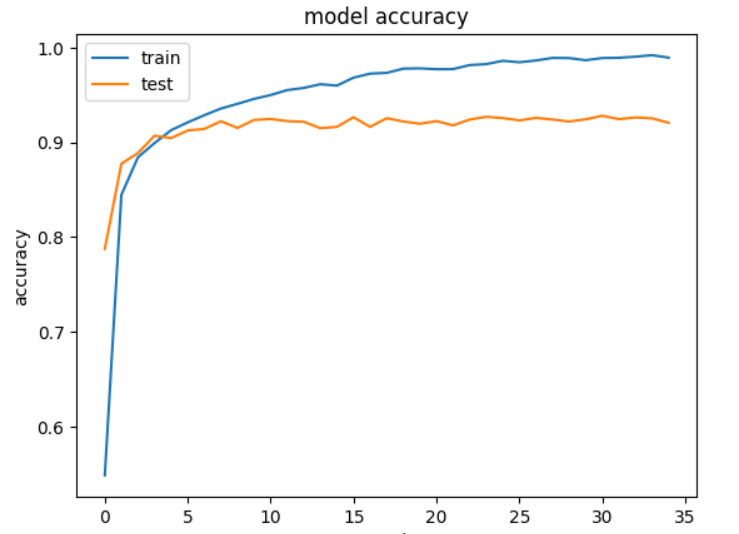
* Sơ đồ hoạt động:

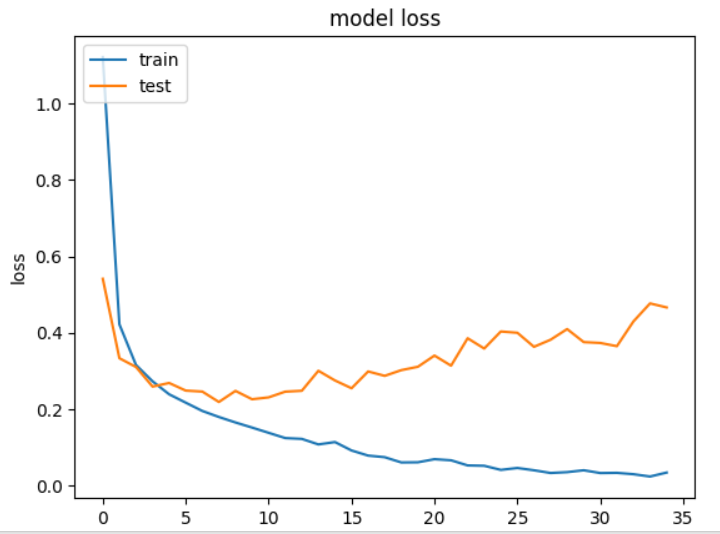


**Params = (k \* k \* input\_channels + 1) \* num\_filters**

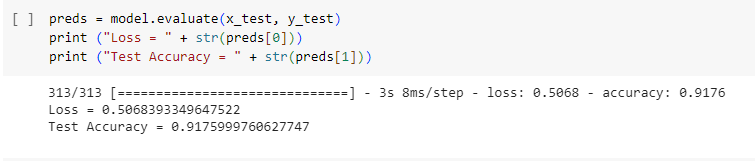


#### **3.3.4: Đồ thị biểu diễn lịch sử**





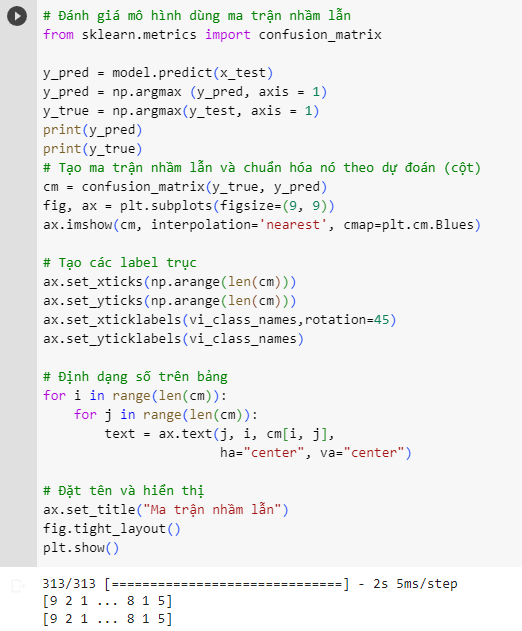
### **3.4. Đánh giá mô hình**

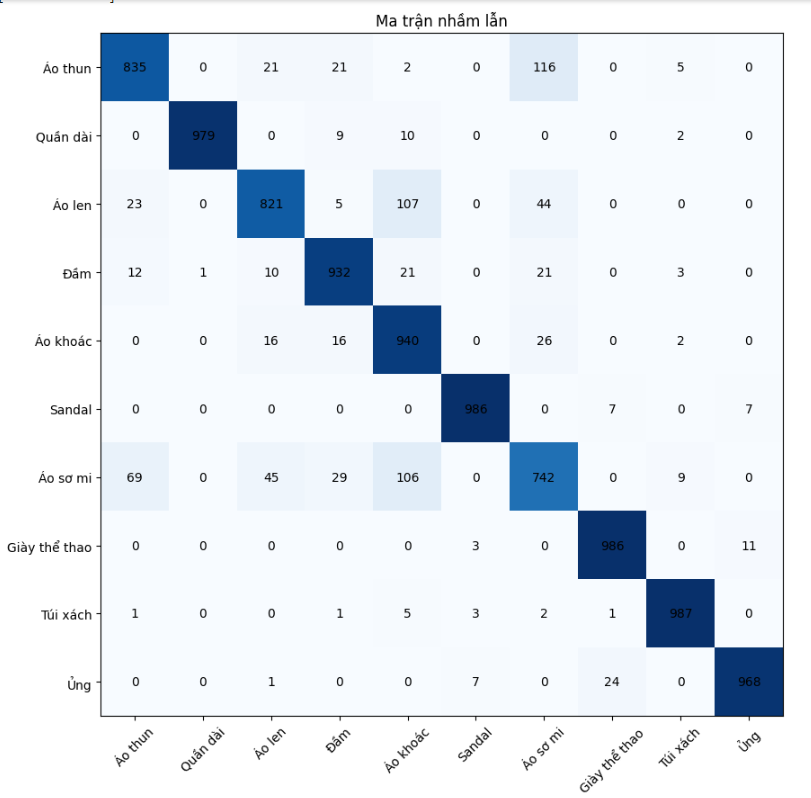


* model.evaluate() thực hiện đánh giá mô hình trên tập dữ liệu kiểm tra.
* x\_test là các dữ liệu kiểm tra, đại diện cho các ảnh trong tập kiểm tra.
* y\_test là nhãn tương ứng với mỗi ảnh trong tập kiểm tra.
* preds[0] chứa giá trị tổn thất của mô hình trên tập dữ liệu kiểm tra.
* preds[1] chứa độ chính xác của mô hình trên tập dữ liệu kiểm tra.

### **3.5. Hiển thị ma trận nhầm lẫn**

#### **3.5.1. Ma trận nhầm lẫn**





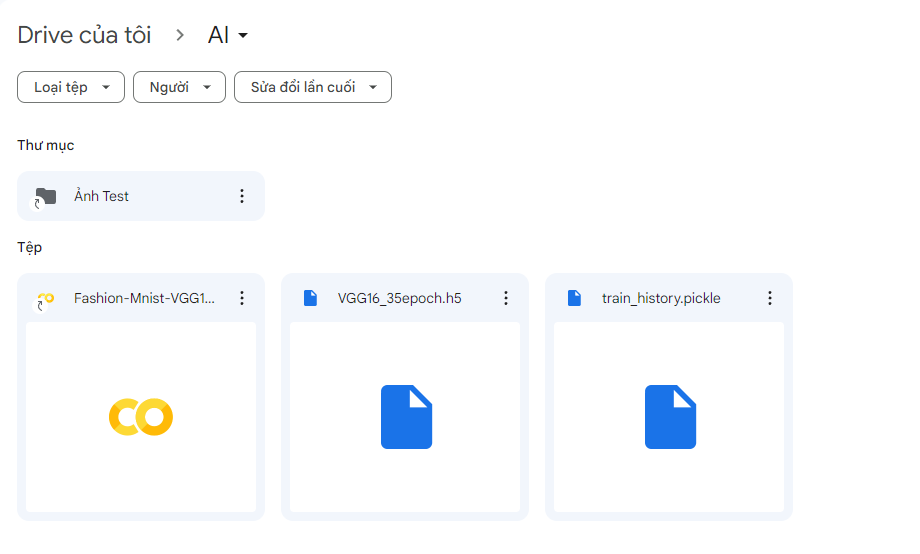
#### **3.5.2. Đánh giá phân tích**

* Dựa vào ma trận nhầm lẫn trên chúng ta thấy túi xách dự đoán chính xác 987/1000 cho biết túi xách ít bị nhầm.
* Áo sơ mi ít chính xác nhất 742/1000 cho biết nhiều khả năng bị nhầm nhất.
* Dự đoán áo thun là áo sơ mi nhiều nhất , 116/1000
* Dự đoán áo len là áo khoác nhiều thứ hai, 107/1000
* Dự đoán áo sơ mi là áo khoác nhiều thứ ba, 106/1000

=> Áo sơ mi, áo thun, áo khoác, áo len rất dễ nhầm lẫn

## **Lưu lịch sử đào tạo và lưu mô hình**

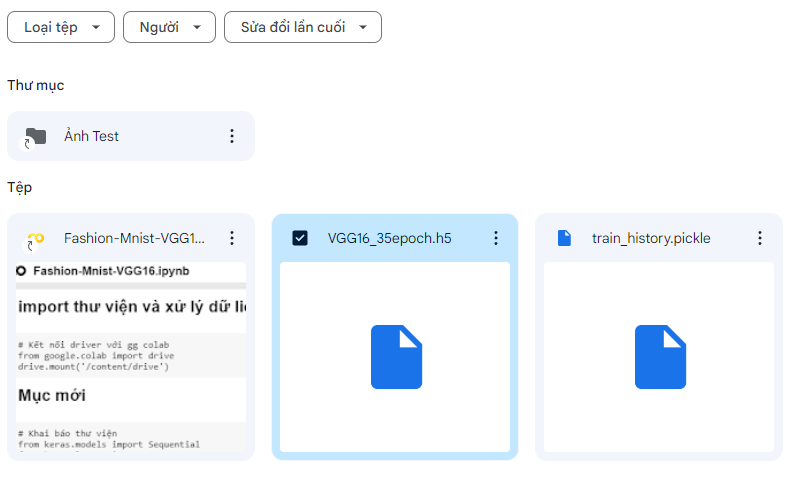
### **4.1. Lưu lịch sử đào tạo**



### **4.2. Lưu mô hình**

* Đào tạo chương trình mất nhiều thời gian, khi chạy các mô hình phức tạp hơn và tập dữ liệu hình ảnh lớn hơn, thường mất mười giờ hoặc vài ngày. Đôi khi máy tính có thể bị sập vì lý do nào đó, vì vậy phần đào tạo trước đó sẽ bị mất. Giải pháp là lưu mô hình sau mỗi lần thực hiện chương trình. Trước khi đào tạo chương trình tiếp theo, tải trọng lượng mô hình trước khi tiếp tục đào tạo.





# **V. Test model với hình ảnh được tải lên từ hệ thống cục bộ.**

