

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐH SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**🕯✡🕮🕮✡🕯**

****

BÁO CÁO ĐỒ ÁN MACHINE LEARNING

**ĐỀ TÀI: PHÂN LOẠI XE HƠI**

|  |  |  |
| --- | --- | --- |
| **GVHD:** | **Thầy Lê Văn Vinh** |  |
| **SVTH:** | **Lữ Phước Hưng** | **16110105** |
|  | **Nguyễn Tấn Sang** | **16110197** |

***TP. Hồ Chí Minh, tháng 7 năm 2020***

**ĐIỂM SỐ**

|  |  |  |  |
| --- | --- | --- | --- |
| **TIÊU CHÍ** | **NỘI DUNG** | **TRÌNH BÀY** | **TỔNG** |
| **ĐIỂM** |  |  |  |

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

#### ……………………………………………………………………………………

TP.HCM, ngày … tháng 7 năm 2020

**Giáo viên hướng dẫn**

##### (ký và ghi họ tên)

#### …………………….....

**LỜI CẢM ƠN**

Để hoàn thành tốt đề tài và bài báo cáo này, chúng tôi xin gửi lời cảm ơn chân thành đến thầy Vinh, người đã trực tiếp hỗ trợ chúng tôi trong cả quá trình làm đề tài. Chúng tôi cảm ơn thầy đã tận tình hướng dẫn chúng tôi từ khâu chọn đề tài, đưa ra những lời khuyên từ kinh nghiệm thực tiễn của mình để định hướng cho chúng tôi hướng đi đúng với đề tài đã chọn, thầy đã luôn theo dõi chúng tôi trong suốt quá trình thực hiện đề tài, luôn giải đáp thắc mắc và đưa ra những góp ý, chỉnh sửa kịp thời giúp chúng tôi khắc phục nhược điểm và hoàn thành tốt đề tài đã chọn. Một lần nữa, chúng tôi xin cảm ơn thầy.

Đề tài và bài báo cáo được chúng tôi thực hiện trong khoảng thời gian ngắn, là khoảng thời gian có hạn với những kiến thức còn hạn chế cùng nhiều điều bỡ ngỡ khác. Do đó, trong quá trình làm nên đề tài có những thiếu sót là điều không thể tránh khỏi nên chúng tôi rất mong nhận được những ý kiến đóng góp của thầy để kiến thức của chúng tôi được hoàn thiện hơn và chúng tôi có thể làm tốt hơn nữa trong những lần sau. Chúng tôi xin chân thành cảm ơn.

Cuối lời, chúng tôi kính chúc thầy luôn dồi dào sức khỏe và thành công hơn nữa trong sự nghiệm trồng người. Một lần nữa chúng tôi xin chân thành cảm ơn.

TP.HCM, ngày … tháng 7 năm 2020

Nhóm sinh viên thực hiện

**MỤC LỤC**

[LỜI NÓI ĐẦU 1](#_bookmark0)

[CHƯƠNG 1: TỔNG QUAN VỀ CÔNG NGHỆ SỬ DỤNG 2](#_bookmark1)

* 1. [Giải thuật CNN 2](#_bookmark2)
     1. [Khái niệm 2](#_bookmark3)
     2. [Cấu trúc mạng CNN 2](#_bookmark4)
     3. [Cách chọn tham số cho CNN 7](#_bookmark5)
     4. [Mô hình VGG16 7](#_bookmark6)
  2. [Công nghệ PyTorch 8](#_bookmark7)
     1. [Khái niệm 8](#_bookmark8)
     2. [Mô hình Pytorch 8](#_bookmark9)
     3. [Ưu điểm 9](#_bookmark10)
     4. [Nhược điểm 9](#_bookmark11)

[CHƯƠNG 2: CHƯƠNG TRÌNH PHÂN LOẠI XE 11](#_bookmark12)

* 1. [Import các thư viện và xác định thiết bị 11](#_bookmark13)
  2. [Truy cập dữ liệu vào google colab trên drive 11](#_bookmark14)
  3. [Load data và tranform data 12](#_bookmark15)
  4. [Khởi tạo hàm train model 13](#_bookmark16)
  5. [Khởi tạo model tính độ chính xác khi test 14](#_bookmark17)
  6. [Khởi tạo model vgg16, loss class, optimizer class 14](#_bookmark18)
  7. [Thực hiện train model 15](#_bookmark19)
  8. [Vẽ plot cho kết quả 16](#_bookmark20)
  9. [Tìm tên của xe 17](#_bookmark21)
  10. [Test model với 1 ảnh bất kì 17](#_bookmark22)
  11. [Hiển thị ảnh và tên của ảnh 17](#_bookmark23)

[KẾT LUẬN 19](#_bookmark24)

[Tài liệu tham khảo 20](#_bookmark25)

**DANH MỤC HÌNH**

Hình 1-1 Cấu trúc mạng CNN 3

Hình 1-2 1. Tạo ra neuron ẩn đầu tiên trong lớp ẩn thứ nhất 4

Hình 1-3 2. Dịch filter qua bên phải một cột sẽ tạo được neuron ẩn thứ 2 4

Hình 1-4 Phân tách dữ liệu ảnh 5

Hình 1-5 Lớp Max Pooling 6

Hình 1-6 Fully connected layer 7

Hình 1-7 Chi tiết các biến thế cấu trúc của các mô hình VGG 7

Hình 2-1 Code các lệnh import thư viện 11

Hình 2-2 Code lệnh truy cập dữ liệu vào colab 11

Hình 2-3 Code load data và tranform data 12

Hình 2-4 Tạo model cho việc train data 13

Hình 2-5 Khởi tạo model tính độ chính xác khi test 14

Hình 2-6 Khởi tạo model vgg16, loss class, optimizer class 14

Hình 2-7 Thực hiện train model 15

Hình 2-8 Vẽ plot cho kết quả 16

Hình 2-9 Tìm tên của xe 17

Hình 2-10 Test model với 1 ảnh bất kỳ 17

Hình 2-11 Code hiển thị kết quả và tên ảnh 17

Hình 2-12 Hình ảnh kết quả 18

# LỜI NÓI ĐẦU

Với xã hội ngày càng phát triển, thì nhu cầu khi con người đạt được mức di chuyển nhanh được thì lại có thêm nhu cầu về vẻ đẹp bề ngoài mà không ngừng cho ra nhiều sản phẩm sẽ hơi ngày càng đẹp mắt hơn và tốt hơn để đáp ứng được nhu cầu của con người. Đồng nghĩa với việc đó thì việc phân loại xe ngày càng khó khăn hơn khi con người có cái nhìn quan sát có giới hạn thì giờ đây chính công nghệ đã giúp cho con người thực hiện được điều đó.

Chương trình phân loại xem hỗ trợ cho con người nhận biết được xem nào loại gì thông qua việc phân tích hình ảnh từ camera ghi nhận được. Đó cũng là lí do chúng tôi chọn đề tài này để thực hiện đồ án của mình.

# CHƯƠNG 1: TỔNG QUAN VỀ CÔNG NGHỆ SỬ DỤNG

## Giải thuật CNN

### Khái niệm

Convolutional Neural Network (CNN – 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. Như hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động. [1]

CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh. Để tìm hiểu tại sao thuật toán này được sử dụng rộng rãi cho việc nhận dạng (detection), chúng ta hãy cùng tìm hiểu về thuật toán này.

### Cấu trúc mạng CNN

[2] Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh 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.

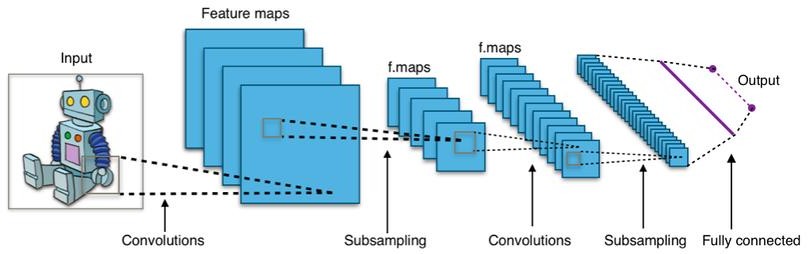
Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw

pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Hình 1-1 Cấu trúc mạng CNN

Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter. Đó là lý do tại sao CNN cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên.

Mạng CNN sử dụng 3 ý tưởng cơ bản:

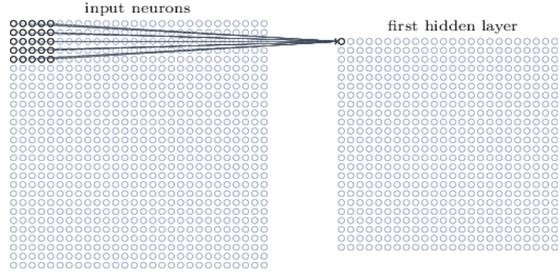
* + - Các trường tiếp nhận cục bộ (local receptive field).
    - Trọng số chia sẻ (shared weights).
    - Tổng hợp (pooling).

###### Trường tiếp nhận cục bộ (local receptive field)

Đầu vào của mạng CNN là một ảnh. Ví dụ như ảnh có kích thước 28×28 thì tương ứng đầu vào là một ma trận có 28×28 và giá trị mỗi điểm ảnh là một ô trong ma trận. Trong mô hình mạng ANN truyền thống thì chúng ta sẽ kết nối các neuron đầu vào vào tầng ảnh.

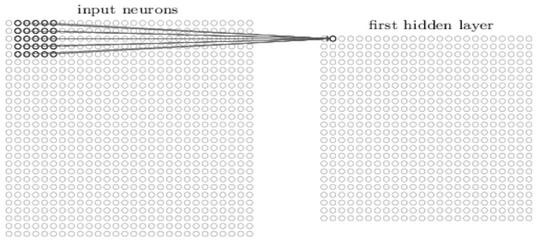
Một cách tổng quan, ta có thể tóm tắt các bước tạo ra 1 hidden layer bằng các cách sau:

###### Tạo ra neuron ẩn đầu tiên trong lớp ẩn thứ nhất



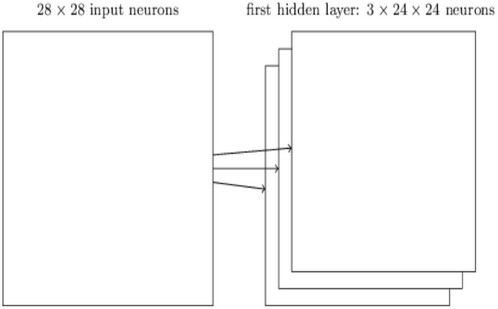
Hình 1-2 1. Tạo ra neuron ẩn đầu tiên trong lớp ẩn thứ nhất

###### Dịch filter qua bên phải một cột sẽ tạo được neuron ẩn thứ 2.



Hình 1-3 2. Dịch filter qua bên phải một cột sẽ tạo được neuron ẩn thứ 2

Với bài toán nhận dạng ảnh người ta thường gọi ma trận lớp đầu vào là feature map, trọng số xác định các đặc trương là shared weight và độ lệch xác định một feature map là shared bias. Như vậy đơn giản nhất là qua các bước trên chúng ta chỉ có 1 feature map. Tuy nhiên trong nhận dạng ảnh chúng ta cần nhiều hơn một feature map.



Hình 1-4 Phân tách dữ liệu ảnh

Như vậy, local receptive field thích hợp cho việc phân tách dữ liệu ảnh, giúp chọn ra những vùng ảnh có giá trị nhất cho việc đánh giá phân lớp.

###### Trọng số chia sẻ (shared weight and bias)

Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-ron trong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Vậy mối quan hệ giữa số lượng Feature map với số lượng tham số là gì?

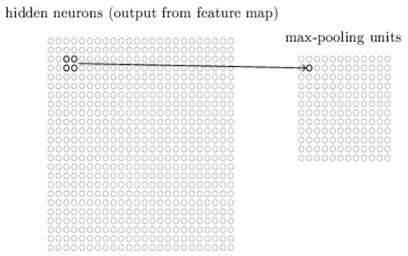
Chúng ta thấy mỗi fearture map cần 25 = 5×5 shared weight và 1 shared bias. Như vậy mỗi feature map cần 5×5+1 = 26 tham số. Như vậy nếu có 10 feature map thì có 10×26 = 260 tham số. Chúng ta xét lại nếu layer đầu tiên có kết nối đầy đủ nghĩa là chúng ta có 28×28=784 neuron đầu vào như vậy ta chỉ có 30 neuron ẩn. Như vậy ta cần 28x28x30 shared weight và 30 shared bias. Tổng số tham số là 28x28x30+30 tham số lớn hơn nhiều so với CNN. Ví dụ vừa rồi chỉ mô tả để thấy được sự ước lượng số lượng tham số chứ chúng ta không so sánh được trực tiếp vì 2 mô hình khác nhau. Nhưng điều chắc chắn là nếu mô hình có số lượng tham số ít hơn thì nó sẽ chạy nhanh hơn.

Tóm lại, một convolutional layer bao gồm các feature map khác nhau. Mỗi một feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN.

###### Lớp tổng hợp (pooling layer)

Lớp pooling thường được sử dụng ngay sau lớp convulational để đơn giản hóa thông tin đầu ra để giảm bớt số lượng neuron.

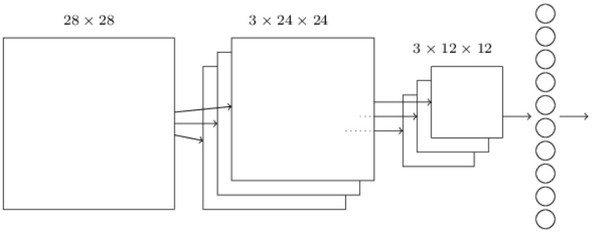
Thủ tục pooling phổ biến là max-pooling, thủ tục này chọn giá trị lớn nhất trong vùng đầu vào 2×2.



Hình 1-5 Lớp Max Pooling

Như vậy qua lớp Max Pooling thì số lượng neuron giảm đi phân nửa. Trong một mạng CNN có nhiều Feature Map nên mỗi Feature Map chúng ta sẽ cho mỗi Max Pooling khác nhau. Chúng ta có thể thấy rằng Max Pooling là cách hỏi xem trong các đặc trưng này thì đặc trưng nào là đặc trưng nhất. Ngoài Max Pooling còn có L2 Pooling.

Cuối cùng ta đặt tất cả các lớp lại với nhau thành một CNN với đầu ra gồm các neuron với số lượng tùy bài toán.



Hình 1-6 Fully connected layer

2 lớp cuối cùng của các kết nối trong mạng là một lớp đầy đủ kết nối (fully connected layer). Lớp này nối mọi nơron từ lớp max pooled tới mọi nơron của tầng ra.

### Cách chọn tham số cho CNN

Số các convolution layer: càng nhiều các convolution layer thì performance càng được cải thiện. Sau khoảng 3 hoặc 4 layer, các tác động được giảm một cách đáng kể.

Filter size: thường filter theo size 5×5 hoặc 3×3.

Pooling size: thường là 2×2 hoặc 4×4 cho ảnh đầu vào lớn.

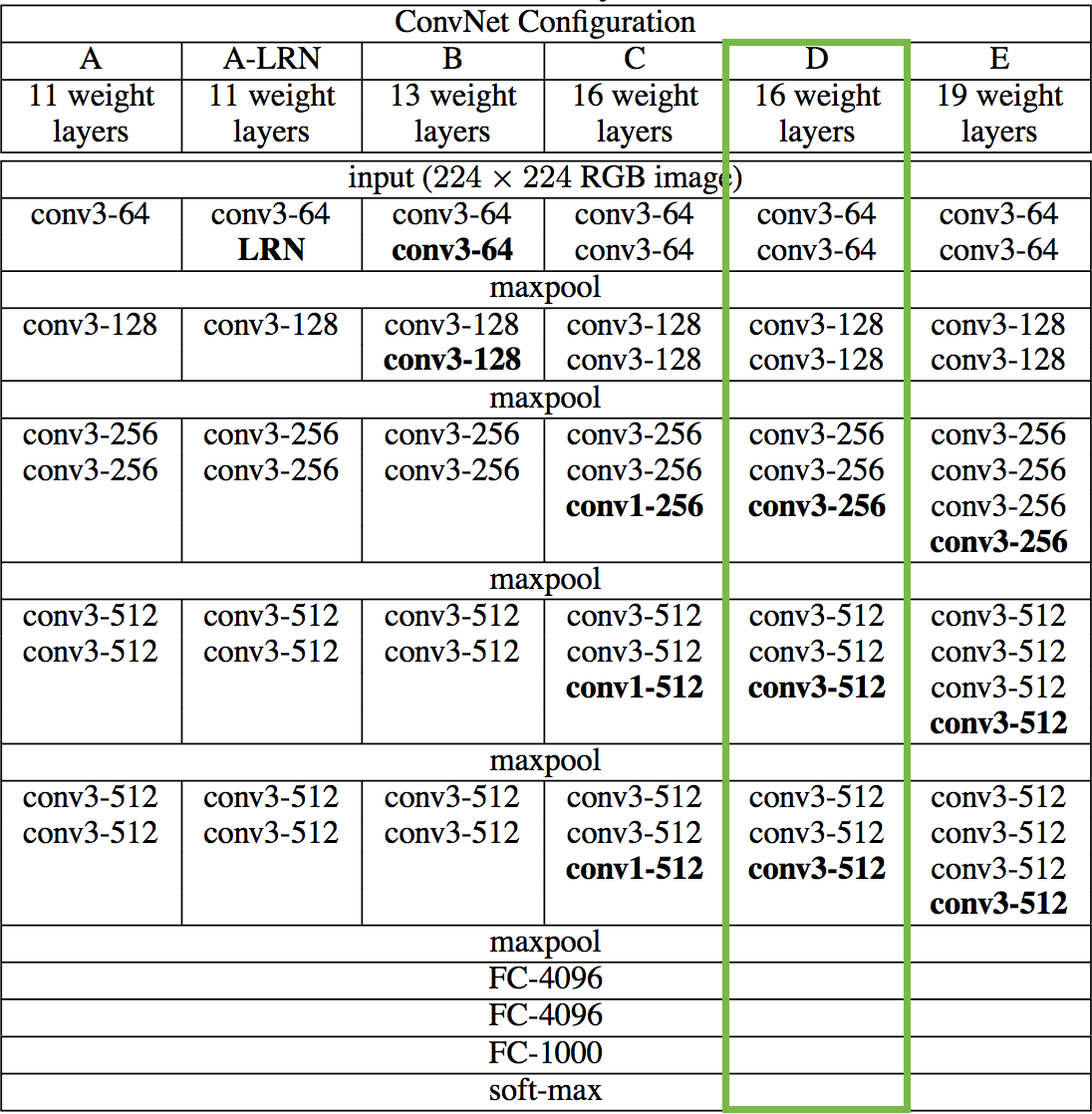
Cách cuối cùng là thực hiện nhiều lần việc train test để chọn ra được param tốt

nhất.

### Mô hình VGG16

Đầu tiên phải kể tới mạng VGG. VGG ra đời năm 2015 và được giới thiệu tại hội

thảo ICLR 2015. Kiến trúc của mô hình này có nhiều biến thể khác nhau: 11 layers, 13 layers, 16 layers, và 19 layers, chi tiết trong hình sau và cụ thể hơn là mô hình VGG16:[3]



Hình 1-7 Chi tiết các biến thế cấu trúc của các mô hình VGG

Nguyên tắc thiết kế của các mạng VGG nói chung rất đơn giản: 2 hoặc 3 layers Convolution (Conv) và tiếp nối sau đó là 1layer Max Pooling 2D. Ngay sau Conv cuối cùng là 1 Flatten layer để chuyển ma trận 4 chiều của Conv layer về ma trận 2 chiều. Tiếp nối sau đó là các Fully-connected layers và 1 Softmax layer. Do VGG được training trên tập dữ liệu của ImageNet có 1000 class nên ở Fully-connected layer cuối cùng sẽ có 1000 units.

Trong Keras hiện tại VGG-16 có 2 params chính là **include\_top (True / False): có sử dụng các Fully-connected layer hay không** và **weights ('imagenet' / None): có sử dụng pre-trained weights của ImageNet hay không**. *# VGG 16*

from keras.applications.vgg16 import VGG16

*# Sử dụng pre-trained weight từ ImageNet và không sử dụng các Fully-connected layer ở cuối*

pretrained\_model = VGG16(include\_top=False, weights='imagenet')

## Công nghệ PyTorch

### Khái niệm

PyTorch là một framework được xây dựng dựa trên python cung cấp nền tảng tính toán khoa học phục vụ lĩnh vực Deep learning. Pytorch tập trung vào 2 khả năng chính:

* Một sự thay thế cho bộ thư viện numpy để tận dụng sức mạnh tính toán của GPU.
* Một platform Deep learning phục vụ trong nghiên cứu, mang lại sự linh hoạt và tốc độ.

### Mô hình Pytorch

###### Load và xử lí data

Đây luôn là một trong những bước đầu tiên trong bất kì một project về Deep Learning nào. Vì thế PyTorch đã hỗ trợ các tiện ích để thực hiện bước này với module torch.utils.data. [4]

2 class quan trọng trong module này là Dataset và DataLoader:

* Dataset được xây dựng trên nền kiểu dữ liệu Tensor và được sử dụng chủ yếu cho các bộ dữ liệu tùy chỉnh.
* DataLoader được sử dụng khi bạn có một tập dữ liệu lớn và bạn muốn tải dữ liệu từ Dataset ở chế độ nền để nó sẵn sàng và chờ sử dụng cho quá trình training loop.

Chúng ta cũng có thể sử dụng torch.nn.DataParallel và torch.distribution trong trường hợp có thể sử dụng song song nhiều máy tính hoặc nhiều GPUs.

###### Xây dựng Neural Network

Module torch.nn được sử dụng để tạo Neural Network. Nó cung cấp tất cả các Neural Network layers phổ biến như các fully connected layers, convolutional layers, các hàm activation và hàm loss, v.v.

Một khi kiến trúc mạng đã được tạo ra và dữ liệu đã sẵn sàng để được feed cho mạng, chúng ta sẽ cần các kỹ thuật để cập nhật các trọng số (weights) và độ lệch (biases) để mạng tiến hành quá trình learning. Các tiện ích này được cung cấp trong module torch.optim. Còn để tính gradient tự động, chúng ta sử dụng module torch.autograd.

###### Model Inference & Compability

Sau khi model đã được train, nó có thể được sử dụng để dự đoán output cho các test cases hoặc thậm chí các datasets mới. Quá trình này được gọi là model inference.

PyTorch cũng cung cấp TorchScript có thể được sử dụng để chạy các model độc lập với Python runtime. Đây có thể được coi là một Máy ảo được thiết kế chủ yếu dành riêng cho việc thao tác với Tensors.

Chúng ta cũng có thể convert model được train bằng PyTorch sang các định dạng như ONNX, cho phép bạn sử dụng các model này trong các Deep Learning framework khác như MXNet, CNTK, Caffe2. Bạn cũng có thể chuyển đổi các model ONNX sang TensorFlow.

###### Làm việc với Tensor

Như đã đề cập ở đầu bài viết, Tensor là 1 mảng n-chiều với các phần tử có cùng kiểu dữ liệu tương tự numpy array. Để hiểu và sử dụng PyTorch Tensors rất dễ dàng.

### Ưu điểm

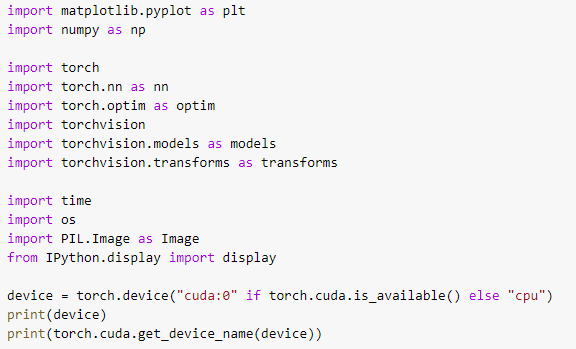
* Mang lại khả năng debug dễ dàng hơn theo hướng interactively, rất nhiều nhà nghiên cứu và engineer đã dùng cả pytorch và tensorflow đều đánh giá cáo pytorch hơn trong vấn đề debug và visualize.
* Hỗ trợ tốt dynamic graphs.
* Được phát triển bởi đội ngũ Facebook.
* Kết hợp cả các API cấp cao và cấp thấp.

### Nhược điểm

* Vẫn chưa được hoàn thiện trong việc deploy, áp dụng cho các hệ thống lớn, ... được như framework ra đời trước nó như tensorflow.
* Ngoài document chính từ pytorch thì vẫn còn khá hạn chế các nguồn tài liệu bên ngoài như các tutorials hay các câu hỏi trên stackoverflow.

# CHƯƠNG 2: CHƯƠNG TRÌNH PHÂN LOẠI XE

## Import các thư viện và xác định thiết bị



Hình 2-1 Code các lệnh import thư viện Import các thư viện sử dụng cho bài toán phân loại xe.

Kiểm tra nếu device nếu là cuda thì available còn không thì dùng cpu Và cho biết thiết bị hiện tại sử dụng.



Cuda hết bộ nhớ nên không sử dụng cuda

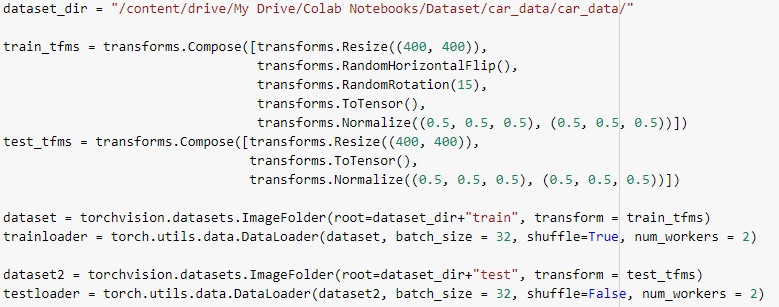
## Truy cập dữ liệu vào google colab trên drive



Hình 2-2 Code lệnh truy cập dữ liệu vào colab

Do chương trình code chạy trên google Colab trên Drive nên ta cần quyền truy cập cái file trên Drive.

## Load data và tranform data

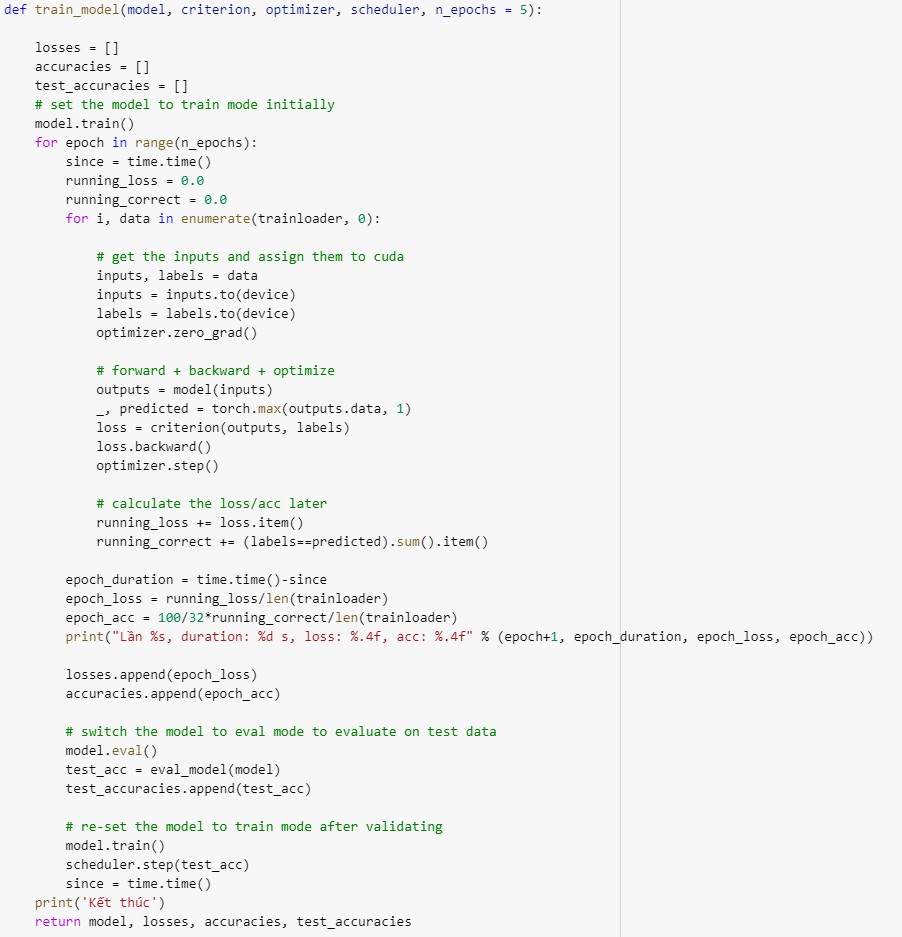


Hình 2-3 Code load data và tranform data Dataser\_dir là đường dẫn đến thư mục chứa data cho việc phân loại

Tiếp đó, ta chuyển đổi dữ liệu train, test và chỉ ra đường dẫn đến thư mục hình ảnh cho train, test.

Cuối cùng sử dụng DataLoader để load dữ liệu hình ảnh và cho train, test

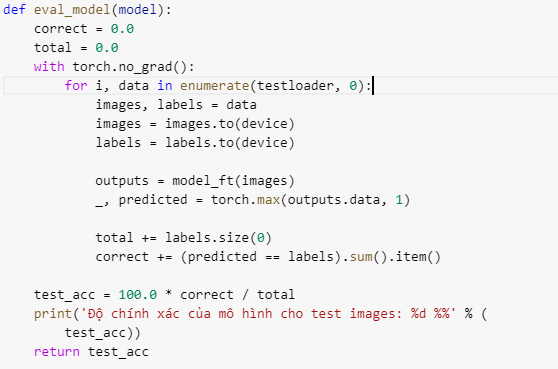
* 1. ***Khởi tạo hàm train model***



Hình 2-4 Tạo model cho việc train data

Train\_model này giúp tính được khoảng thời gian, data không tìm thấy, và độ chính xác cho từng lần train

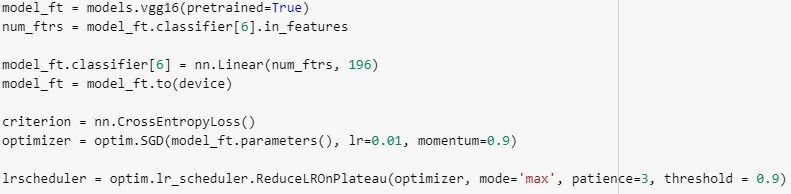
## Khởi tạo model tính độ chính xác khi test



Hình 2-5 Khởi tạo model tính độ chính xác khi test

Sau khi tính toán xong sẽ in ra kết quả phần trăm độ chính xác của dữ liệu test

## Khởi tạo model vgg16, loss class, optimizer class



Hình 2-6 Khởi tạo model vgg16, loss class, optimizer class Nhóm chúng em chọn model để thực hiện phân loại ảnh là VGG16

Và chọn chuẩn hóa classifer [6] với in\_features**=** 4096, out\_features**=**1000

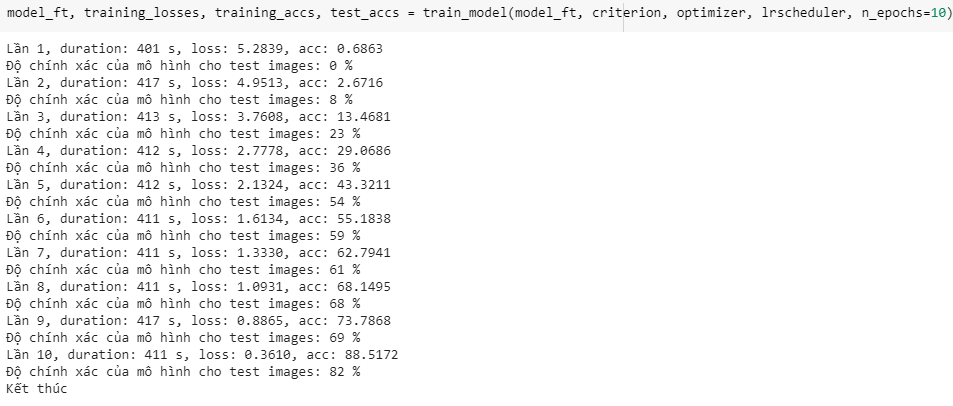
Ta sử dụng class nn.Linear với dữ liệu cho từng đầu vào là 4096 và số lượng nhãn

là 196.

class CrossEntropyLoss() để cân bằng các dữ liệu đầu vào. class optim.SGD để tối ưu hóa các tham số của model\_ft.

Torch.optim.lr\_scheduler cung cấp một số phương pháp để điều chỉnh tốc độ học dựa trên số lượng epochs. Class [ReduceLROnPlateau](https://pytorch.org/docs/stable/optim.html#torch.optim.lr_scheduler.ReduceLROnPlateau)() cho phép giảm tốc độ học tập năng động dựa trên một số phép đo xác nhận.

## Thực hiện train model

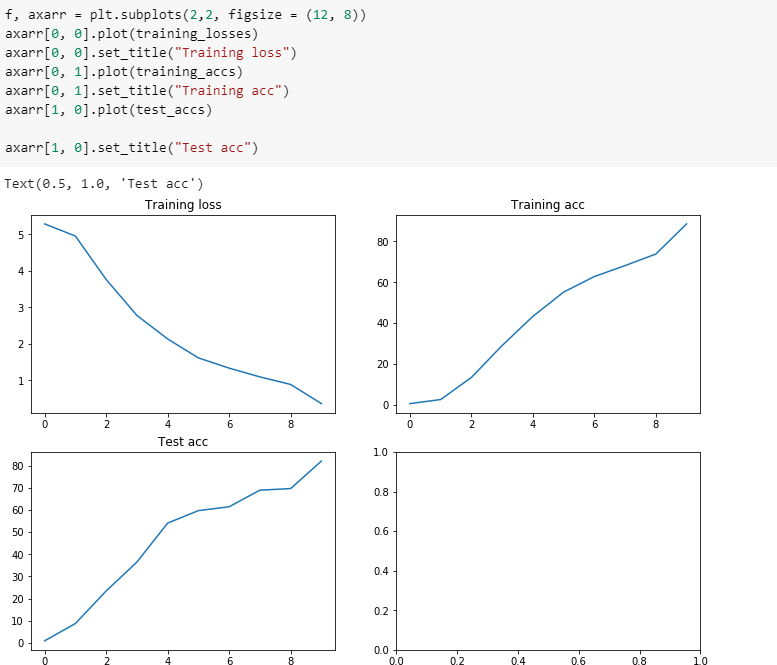


Hình 2-7 Thực hiện train model

Sau khi đã chuẩn bị bắt đầu train cho model train\_model đào tạo lại mô hình mới. Đối với mỗi epochs, nhóm chạy cả dữ liệu đào tạo và xác nhận. Chúng tôi theo dõi mô hình tốt nhất với độ chính xác xác thực cao nhất và trả lại mô hình tốt nhất sau 10 epoch.

Kết quả rất tốt của mô hình VGG16 khi thực hiện 10 lần đào tạo và ở lần thứ 10 có độ chính xác khi thực hiện test là **82%** cho phân loại ảnh

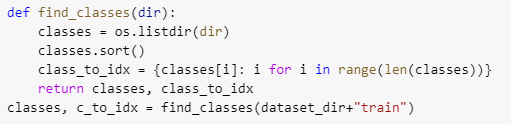
## Vẽ plot cho kết quả



Hình 2-8 Vẽ plot cho kết quả

Vẽ plot thể hiện biểu đồ của Training loss, Training acc và Test acc. Ta có thể thấy mỗi lần train mới sẽ cho kết quả tốt hơn lần trước.

## Tìm tên của xe



Hình 2-9 Tìm tên của xe

## Test model với 1 ảnh bất kì



Hình 2-10 Test model với 1 ảnh bất kỳ

## Hiển thị ảnh và tên của ảnh



Hình 2-11 Code hiển thị kết quả và tên ảnh

Kết quả:





Hình 2-12 Hình ảnh kết quả

# KẾT LUẬN

**Đánh giá:** Nhóm đã hoàn thành thành công ứng dụng phân loại xe hơi với tỉ lệ phân loại chính xác các loại xe trên tập dataset đát mức khá cao, cũng như nghiên cứu được Pytorch và công nghệ CNN (cụ thể ở đây là VGG16) để phục vụ cho việc phân loại xe hơi.

###### Ưu điểm:

* Có độ chính xác khi phân loại xe hơi khác cao hơn 82%.
* Có tập dataset tương đối phù hợp.

###### Nhược điểm:

* Chưa tạo được giao diện thân thiện với người dùng.
* Còn thực hiện ở mức cơ bản.

# Tài liệu tham khảo

1. <https://nguyenvanhieu.vn/thuat-toan-cnn-trong-nlp/>
2. <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-> nlp/Viblo
3. [https://techblog.vn/gioi-thieu-ve-cac-pre-trained-models-trong-linh-vuc-computer- vision](https://techblog.vn/gioi-thieu-ve-cac-pre-trained-models-trong-linh-vuc-computer-vision)
4. [https://medium.com/@haophan\_cs/pytorch-c%C6%A1-b%E1%BA%A3n- gi%E1%BB%9Bi-thi%E1%BB%87u-v%E1%BB%81-pytorch-cda2b15086c](https://medium.com/%40haophan_cs/pytorch-c%C6%A1-b%E1%BA%A3n-gi%E1%BB%9Bi-thi%E1%BB%87u-v%E1%BB%81-pytorch-cda2b15086c)