



ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY

한-베정보통신기술대학교

Chương 5-3. Phân lớp ảnh (Image Classification)

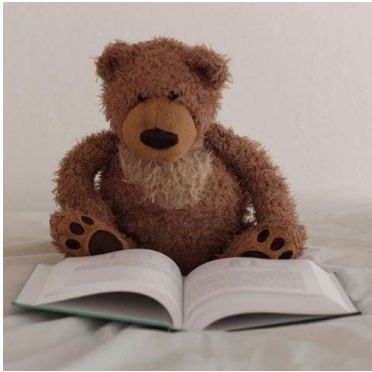
TS. PHẠM NGUYỄN MINH NHỰT

Email: pnmnhut@vku.udn.vn

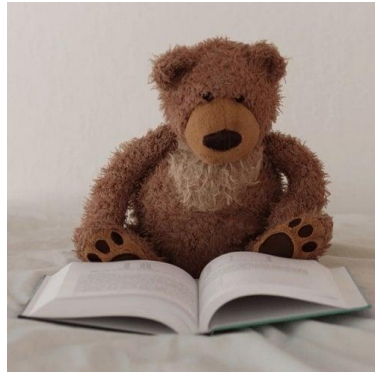
Phone: 0903.501.421

- Tăng kích thước mô hình (Tăng số lớp tích chập) → càng nhiều tham số → mô hình sẽ học được nhiều
- Thay đổi activation function (ví dụ: Tanh, ReLU, sigmoid, LeakyReLU...)
- Thay đổi các tham số: tốc độ học, kích thước ảnh đầu vào, số lượng epoch...
- Thay đổi thuật toán tối ưu (Adam, SGD, RMSprop...)
- Tăng số lượng dữ liệu train → Nếu không đủ sử dụng kỹ thuật tăng cường dữ liệu (Data Augmentation)
- Kích thước Mini - batch nhỏ cũng ảnh hưởng độ chính xác → kích thước thường 32
- Thay đổi kiến trúc mô hình CNN...

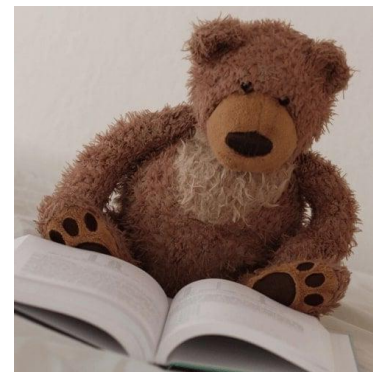
- Khi có ít dữ liệu cho việc training → Data Augmentation là kỹ thuật tạo ra dữ liệu để training từ dữ liệu đang có



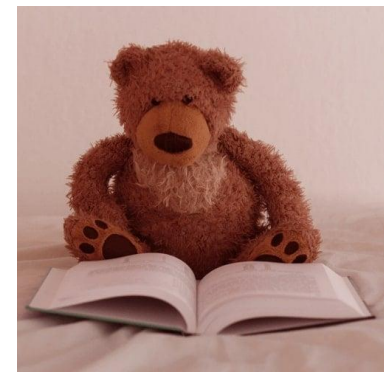
Hình gốc



Lật



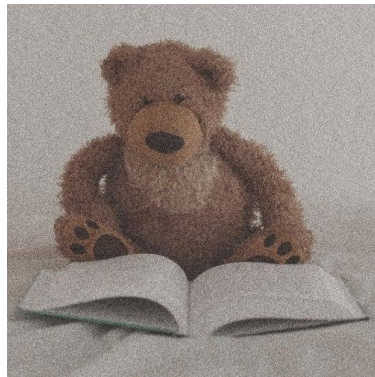
Xoay



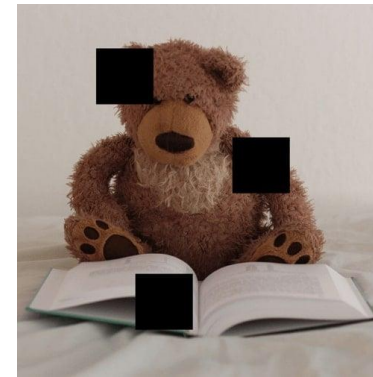
Dịch chuyển màu



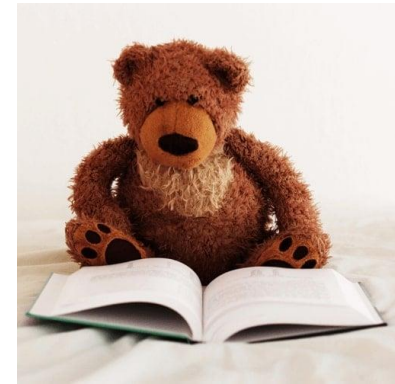
Crop ngẫu nhiên



Thêm nhiễu



Mất mát thông tin

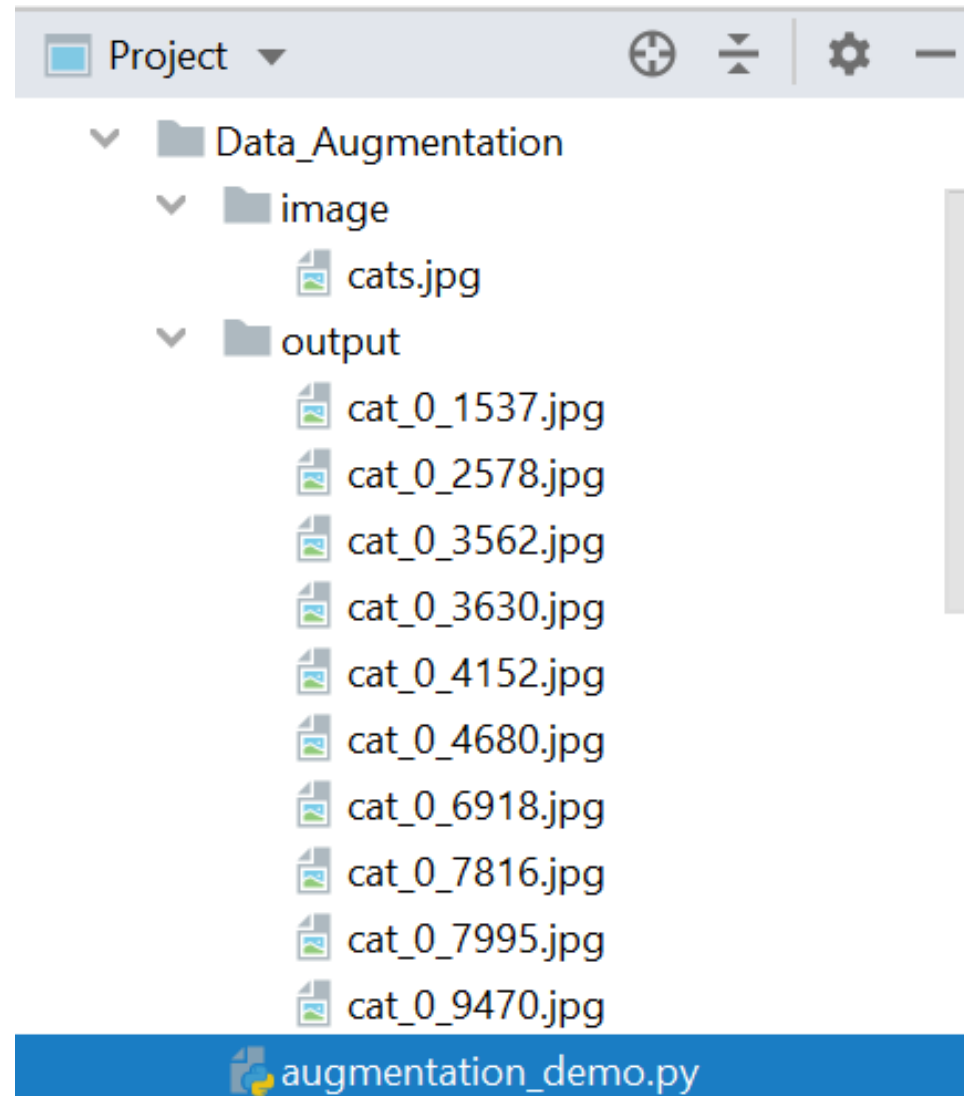


Thay đổi độ tương phản

- Sử dụng lệnh tăng cường dữ liệu trong Keras

```
Data_Aug = ImageDataGenerator(  
    rescale=1./255,           #Thay đổi tỷ lệ  
    rotation_range=40,        #Xoay  
    width_shift_range=0.2,    # Thay đổi chiều rộng  
    height_shift_range=0.2,   # Thay đổi chiều cao  
    shear_range=0.2,          # Xén ảnh  
    zoom_range=0.2,           # Zoom ảnh  
    horizontal_flip=True,     # Lật ảnh  
    fill_mode='nearest')
```

- Áp dụng để tạo thêm Dataset ảnh hoặc dùng trực tiếp trong chương trình khi train



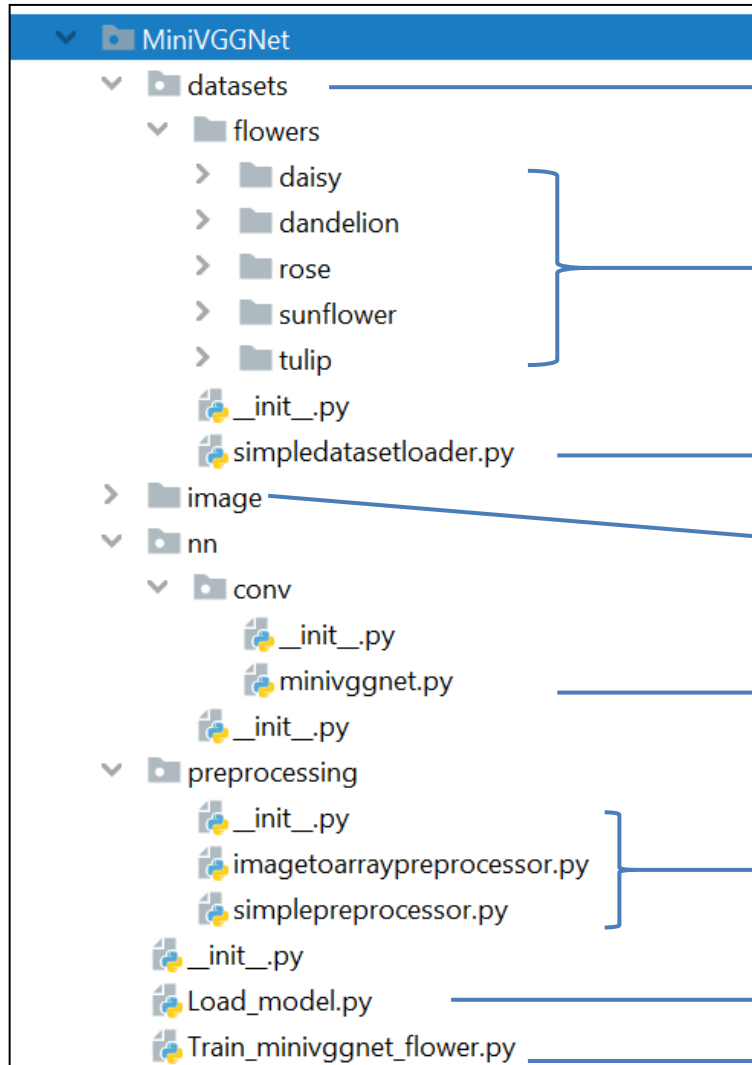
- VGGNet là CNN sử dụng đối với tập dữ liệu ảnh quy mô lớn
- Được giới thiệu bởi Simonyan và Zisserman vào năm 2014
- VGGNet sử dụng kernel kích thước 3×3 trong toàn bộ kiến trúc.
- VGGNet có nhiều phiên bản
 - VGGNet16 → sử dụng 16 layer
 - VGGNet19 → sử dụng 19 layer
 - ...

- MiniVGGNet là một phiên bản nhỏ hơn của VGGNet được \rightarrow có thể dễ dàng triển khai trên máy tính hạn chế tài nguyên
- Kiến trúc MiniVGGNet:
 - Gồm 02 chuỗi lớp: CONV \Rightarrow RELU \Rightarrow CONV \Rightarrow RELU \Rightarrow POOL
 - Theo sau bởi FC \Rightarrow RELU \Rightarrow FC \Rightarrow SOFTMAX
 - Hai lớp CONV đầu tiên sẽ học thông qua 32 bộ lọc, mỗi bộ lọc có kích thước 3×3
 - Hai lớp CONV thứ hai sẽ học thông qua 64 bộ lọc, mỗi bộ lọc có kích thước 3×3
 - Các lớp POOL sử dụng Max Pooling với cửa sổ kích thước 2×2 và Stride (trượt) là 2×2
 - Sử dụng lớp Batch Normalization sau khi activations \rightarrow tránh overfitting và tăng độ chính xác
 - Lớp Dropout thực hiện sau khi POOL và lớp FC
- \rightarrow Dataset tham khảo từ: <https://www.kaggle.com/alxmamaev/flowers-recognition>

- Cấu trúc MiniVGGNet

Layer Type	Output Size	Filter Size / Stride
INPUT IMAGE	$32 \times 32 \times 3$	
CONV	$32 \times 32 \times 32$	$3 \times 3, K = 32$
ACT	$32 \times 32 \times 32$	
BN	$32 \times 32 \times 32$	
CONV	$32 \times 32 \times 32$	$3 \times 3, K = 32$
ACT	$32 \times 32 \times 32$	
BN	$32 \times 32 \times 32$	
POOL	$16 \times 16 \times 32$	2×2
DROPOUT	$16 \times 16 \times 32$	
CONV	$16 \times 16 \times 64$	$3 \times 3, K = 64$
ACT	$16 \times 16 \times 64$	
BN	$16 \times 16 \times 64$	
CONV	$16 \times 16 \times 64$	$3 \times 3, K = 64$
ACT	$16 \times 16 \times 64$	
BN	$16 \times 16 \times 64$	
POOL	$8 \times 8 \times 64$	2×2
DROPOUT	$8 \times 8 \times 64$	
FC	512	
ACT	512	
BN	512	
DROPOUT	512	
FC	10	
SOFTMAX	10	

- Tổ chức project



Chứa Dataset cho Training, Validation, Testing

05 class

Định nghĩa phương thức nạp dữ liệu

Chứa ảnh để thử nghiệm model

Định nghĩa model và layers

Định nghĩa phương thức chuẩn bị dữ liệu trước khi nạp vào model để train

Nạp model (*.hdf5) để viết ứng dụng

Train và đánh giá model



ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY

한-베정보통신기술대학교

Kiểm tra

1: Project

- Project
 - Du_thi
 - MiniVGGNet
 - MiniVGGNet_Animals_KiemTra
 - datasets
 - animals
 - cat
 - dog
 - panda
 - _init_.py
 - simplifiedatasetloader.py
 - image
 - nn
 - preprocessing
 - _init_.py
 - Load_model.py
 - miniVGGNet_animals.hdf5
 - Train_minivggnet_animals.py

```

50 # - momentum: Hệ số quán tính
51 # - nesterov = True: sử dụng phương pháp tối ưu Nesterov accelerated gradient
52 optimizer = SGD(learning_rate=0.01, decay=0.01 / 40, momentum=0.9, nesterov=True)
53 model = MiniVGGNet.build(width=32, height=32, depth=3, classes=3)
54 model.compile(loss="categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
55
56 # Bước 3: Train the network
57 print("[INFO]: Đang training....")
58 H = model.fit(trainX, trainY, validation_data=(testX, testY), batch_size=32, epochs=60, v
59
60 model.save("miniVGGNet_animals.hdf5") # Lưu model
    
```

Terminal: Local x Local (2) x Local (3) x Local (4) x +

cat	0.67	0.64	0.65	262
dog	0.62	0.60	0.61	249
panda	0.85	0.90	0.87	239
accuracy			0.71	750
macro avg	0.71	0.72	0.71	750
weighted avg	0.71	0.71	0.71	750

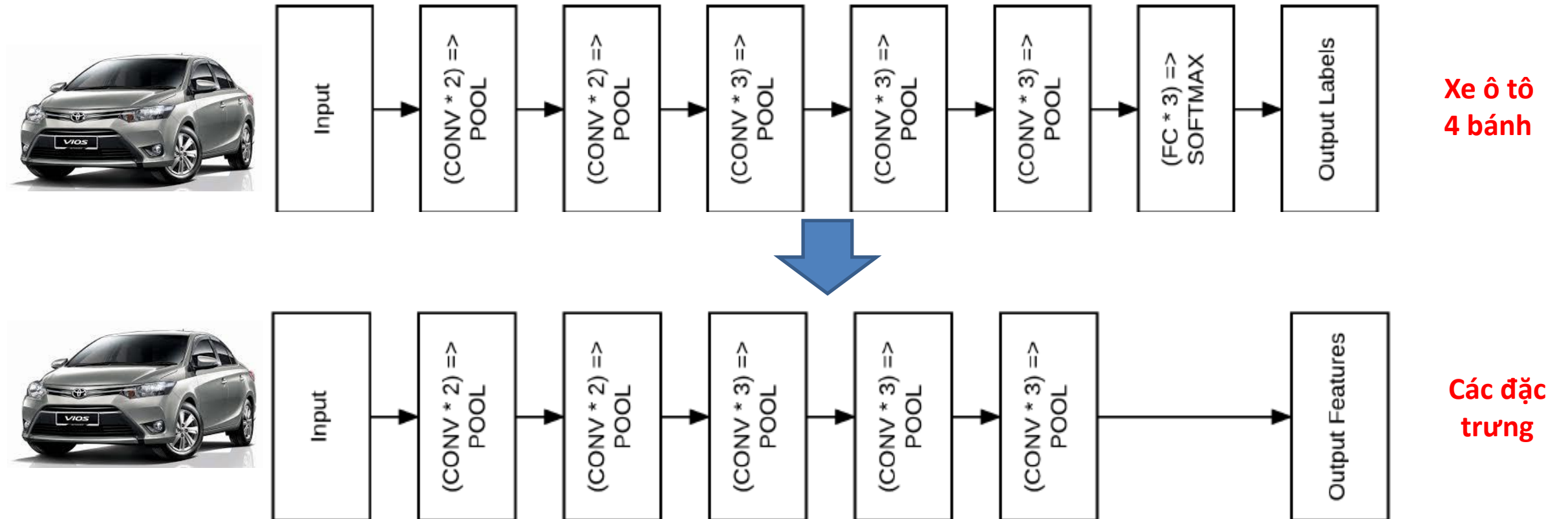
2: Favorites

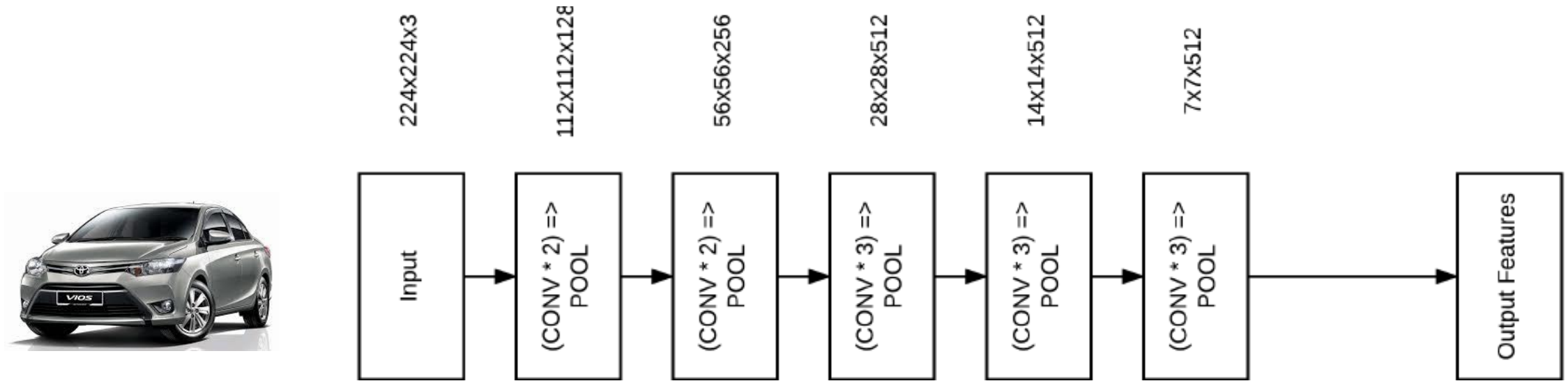
- **Tham khảo MiniVGGNet để tạo model phân lớp animals, với yêu cầu:**
 - Dữ liệu animals: (cat,dog, panda) = (1000,1000,1000)
 - Sử dụng kỹ thuật tăng cường dữ liệu, sao cho (cat,dog, panda) = (1500,1500,1500)
 - Cố định số epoch=60
 - Có thể hiệu chỉnh cấu trúc model, thay đổi tốc độ học...
- **Thứ 5 kiểm tra: 13h00 học thì 13h30 báo cáo kết quả → train trước khi vào học**
- **Điểm:**

Độ chính xác	< 70%	71%	> 71% <= 75%	>75% <=80%	>80% <=90%	>90%
Điểm	0	5	8	8.5	9	10

- **Học chuyển tiếp (transfer learning)** khả năng sử dụng model được đào tạo trước (pre-trained) để tiếp tục học từ dữ liệu mới (dữ liệu mới này chưa được dùng để train cho model trước đó).
- **Ví dụ:**
 - Trước đây đã tạo mạng CNN để nhận diện chó, mèo và gấu trong ảnh
 - Sử dụng Học chuyển tiếp là dùng mạng CNN này để tiếp tục train nhằm nhận diện gấu xám, gấu bắc cực và gấu trúc khổng lồ (các dữ liệu này chưa được xử dụng để train cho model trước đó)
- **Có 2 hình thức học chuyển tiếp khi áp dụng cho deep learning trong thị giác máy tính:**
 - Phương pháp 1: Thay đổi mục đích của mạng CNN đã có → để trích xuất đặc trưng. **Phương pháp này gọi là Trích xuất đặc trưng với mạng Pre-Trained CNN**
 - Phương pháp 2: Thay thế các lớp FC đã có của mạng CNN bằng các lớp FC mới (gắn vào vùng Top của CNN) và tinh chỉnh (fine-tuning) các trọng số của CNN. **Phương pháp này gọi là Tinh chỉnh (fine-tuning) các trọng số của CNN**

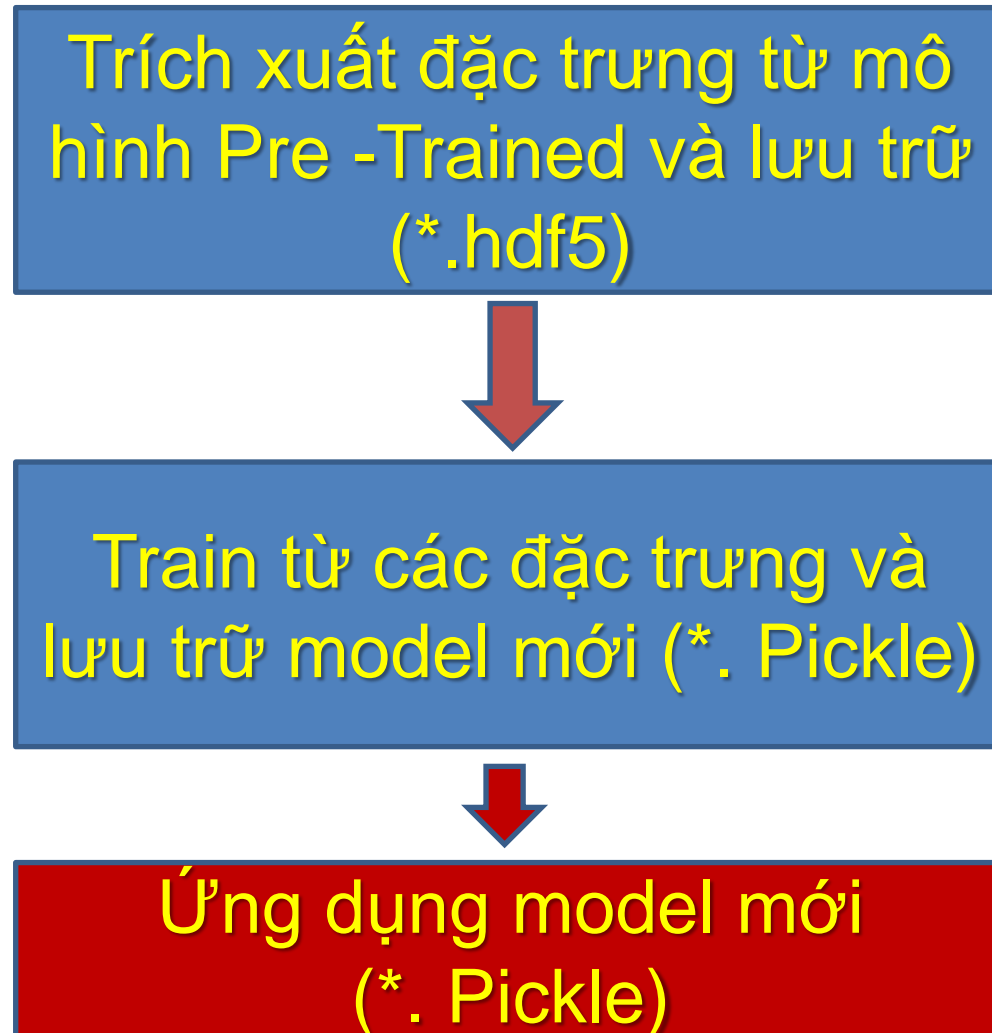
- Xét một pre-trained CNN (VGG16) cho bài toán phân lớp ảnh đã được train trước đó
- Loại bỏ lớp FC và không sử dụng hàm phân lớp (softmax) ở cuối mạng



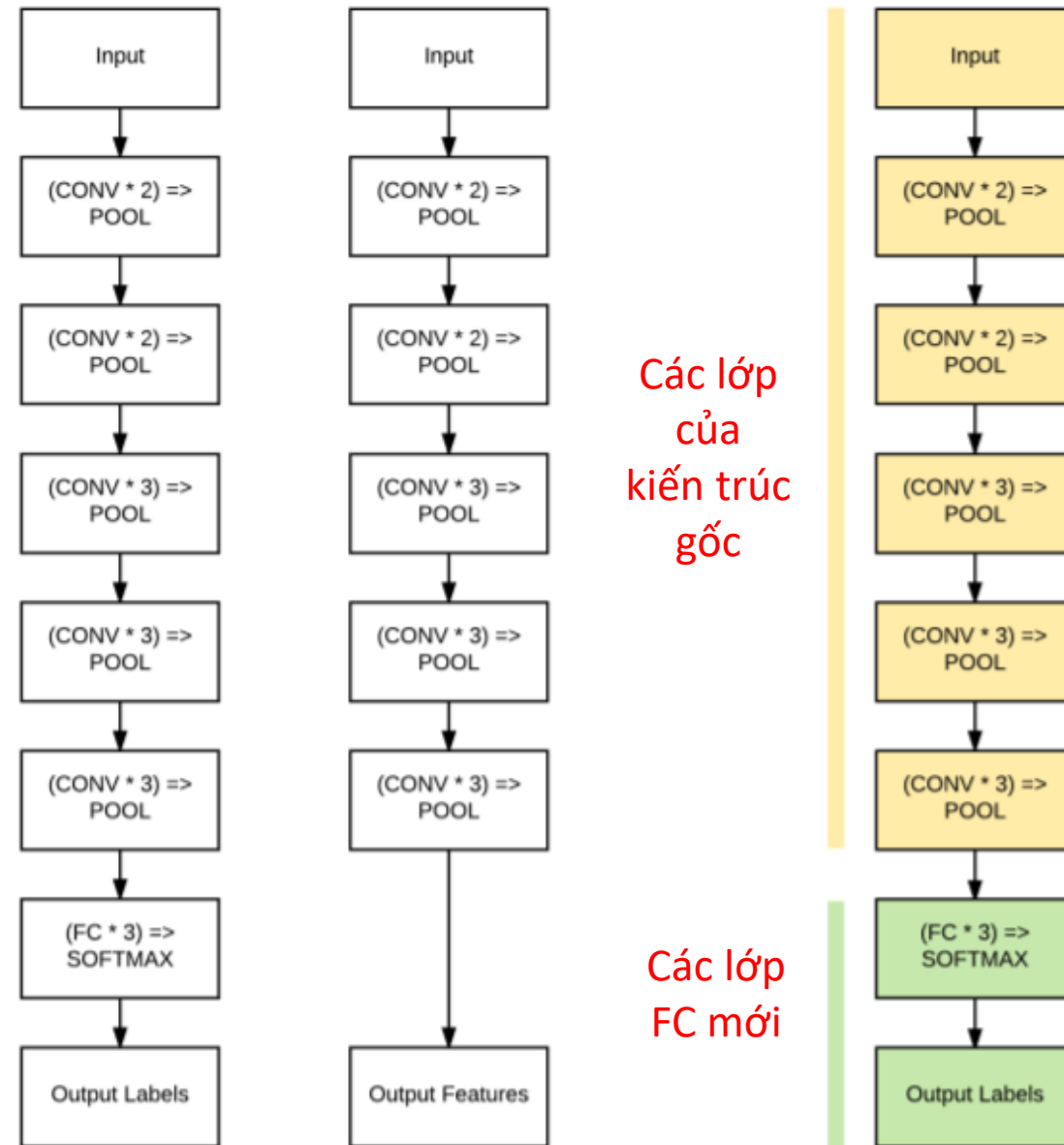


- → Lớp cuối cùng trong mạng có 512 bộ lọc mỗi bộ có kích thước 7×7 .
- → Có véc tơ với $7 \times 7 \times 512 = 25.088$ phần tử → véc tơ đặc trưng biểu diễn nội dung của ảnh.
- Từ đó sử dụng các đặc trưng này để train các mô hình khác → tạo ra các bộ phân lớp để nhận dạng các lớp ảnh mới khác với độ chính xác cao hơn

- Các bước thực hiện theo Phương pháp 1: Trích xuất đặc trưng và train



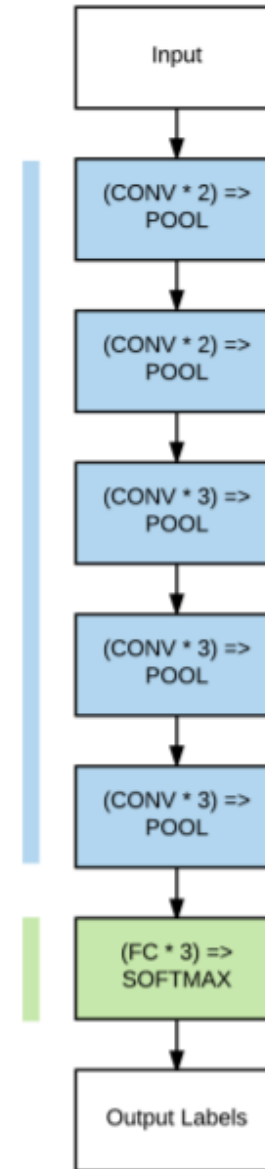
- Xóa phần đầu khỏi mạng, giống như trong trích xuất đặc trưng
- Xây dựng lớp FC mới
- Đặt lớp FC này trên đầu của kiến trúc gốc
- Train model với lớp FC mới này



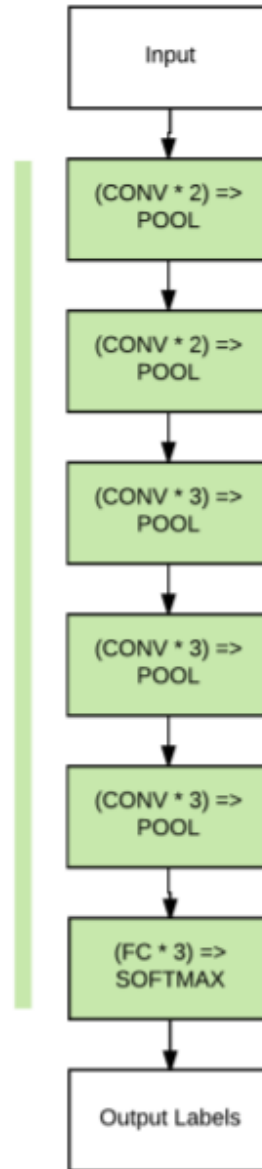
- Thông thường trước khi train, đóng băng các lớp của mô hình gốc (gọi là lớp Base)
- Train lớp FC đến trạng thái “ấm lên”
- Sau đó, mở đóng băng và tinh chỉnh (train) các tham số của mạng

Các lớp bị đóng băng

Chỉ train lớp FC để các lớp chuyển trạng thái “ấm lên”



Mở đóng băng và train tất cả



- **Thực hành:** Sử dụng phương pháp tinh chỉnh (Fine-Tuning) đối với model VGG16
- **Chú ý:** Sử dụng kỹ thuật tăng cường ảnh bằng dòng lệnh của Keras

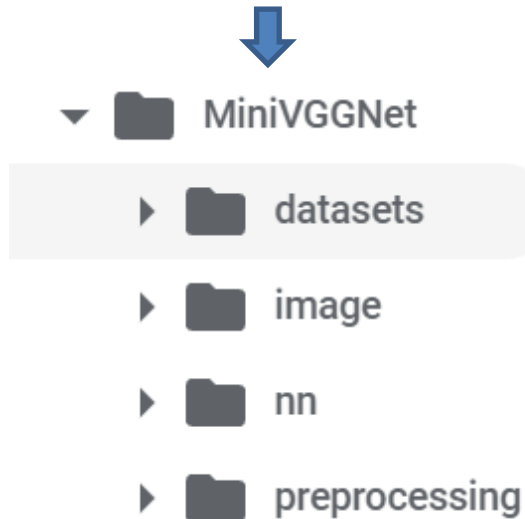
```
aug = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')
```

- VGG16 là mạng CNN được đề xuất bởi K. Simonyan and A. Zisserman, University of Oxford.
- Model sau khi train bởi mạng VGG16 đạt độ chính xác 92.7%
- Dữ liệu [ImageNet](#) gồm 14 triệu hình ảnh thuộc 1000 lớp khác nhau.

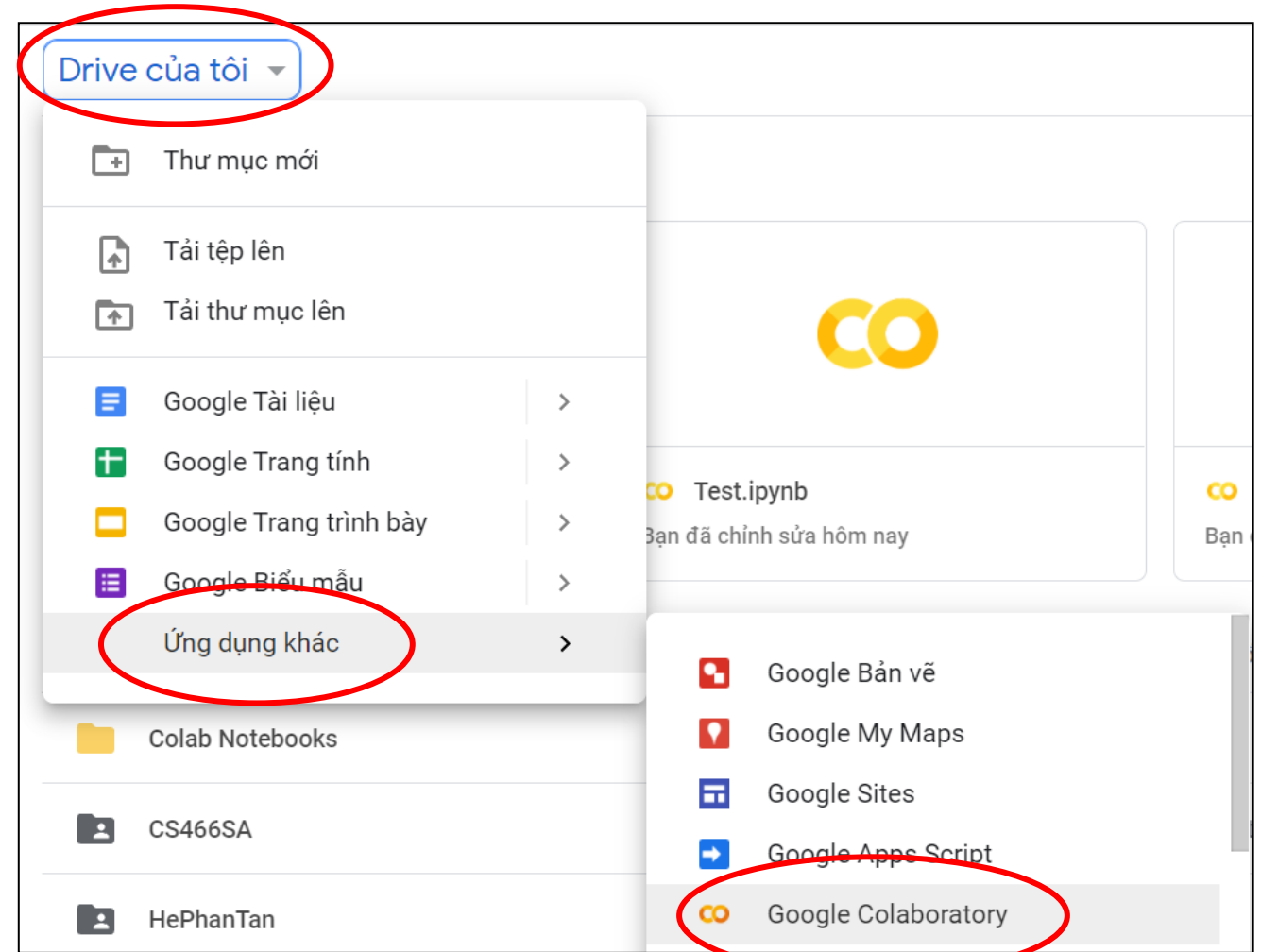


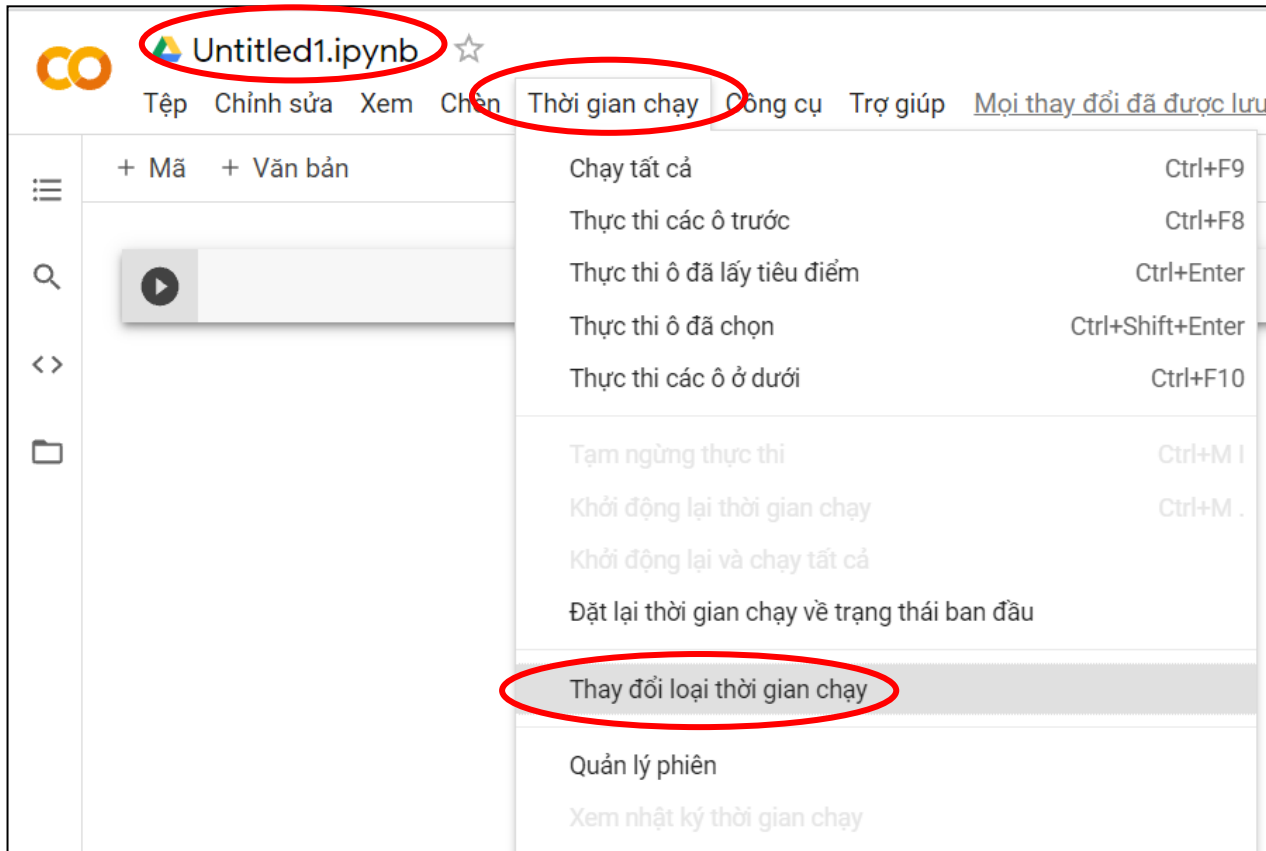
- Colaboratory hay còn gọi là Google Colab là một sản phẩm từ Google Research
- Cho phép chạy các code viết bằng python thông qua trình duyệt
- Ứng dụng các bài toán Data Analysis, Machine Learning, Deep Learning...
- Cung cấp tài nguyên máy tính → GPU và TPU
- Đối với dịch vụ miễn phí giới hạn thời gian sử dụng, tối đa lên tới 12 giờ → Do vậy, kết hợp Drive Google để lưu tài nguyên

- Bước 1: Đăng nhập Drive Google
- Bước 2: Sao chép cấu trúc dự án (trên máy cá nhân) lên Drive Google

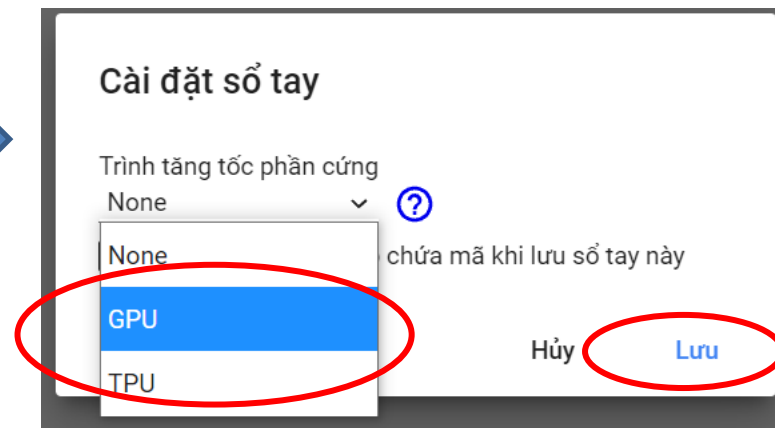


- Bước 3: Chọn Drive của tôi → Ứng dụng khác → Google Colaboratory

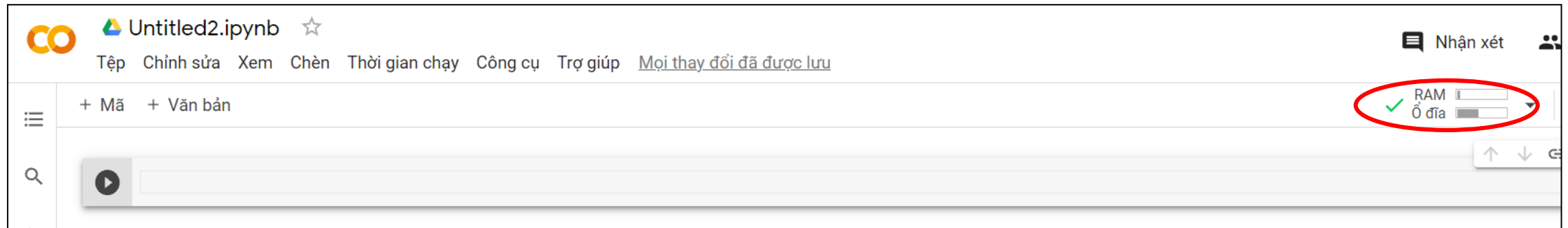




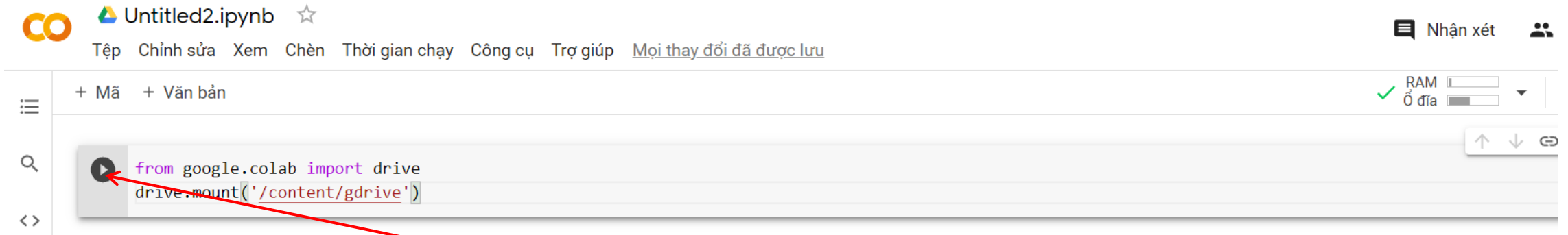
- Bước 4: Đặt tên file: Chọn tại Untitled1.ipynb và nhập tên file mới
- Bước 5: Chọn cấu hình máy tính trên Colab: Chọn Thời gian chạy → Thay đổi loại thời gian chạy → Chọn loại CPU, GPU hoặc TPU



- Bước 6: Thực hiện kết nối đến với cloud của Google → nhấn Kết nối



- Bước 7: Kết nối Colab với Drive Google → Nhập 02 lệnh



CO Untitled2.ipynb ☆

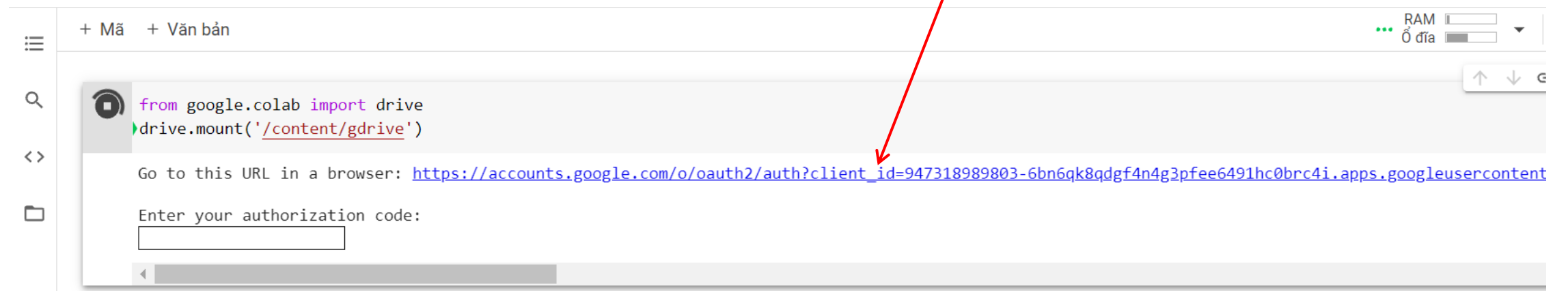
Tệp chỉnh sửa Xem Chèn Thời gian chạy Công cụ Trợ giúp Mọi thay đổi đã được lưu

+ Mã + Văn bản

✓ RAM Ổ đĩa

```
from google.colab import drive
drive.mount('/content/gdrive')
```

- Bước 8: Nhấn nút thực thi lệnh (hình Tam giác) → Nhấn link



+ Mã + Văn bản

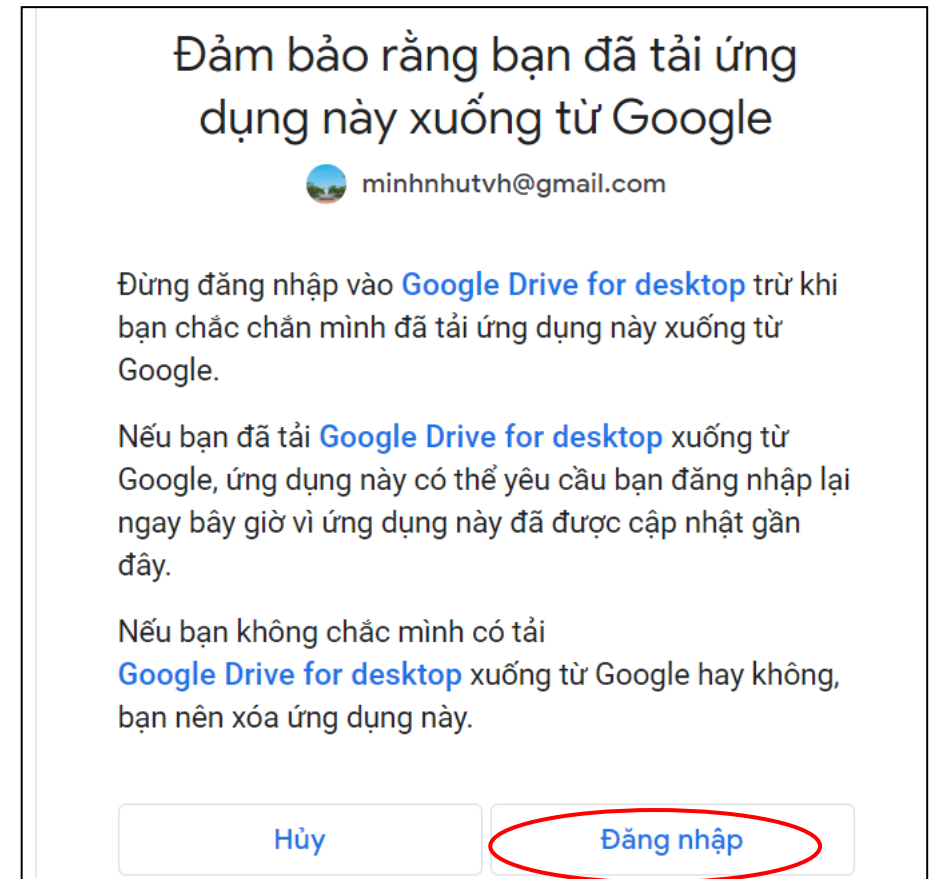
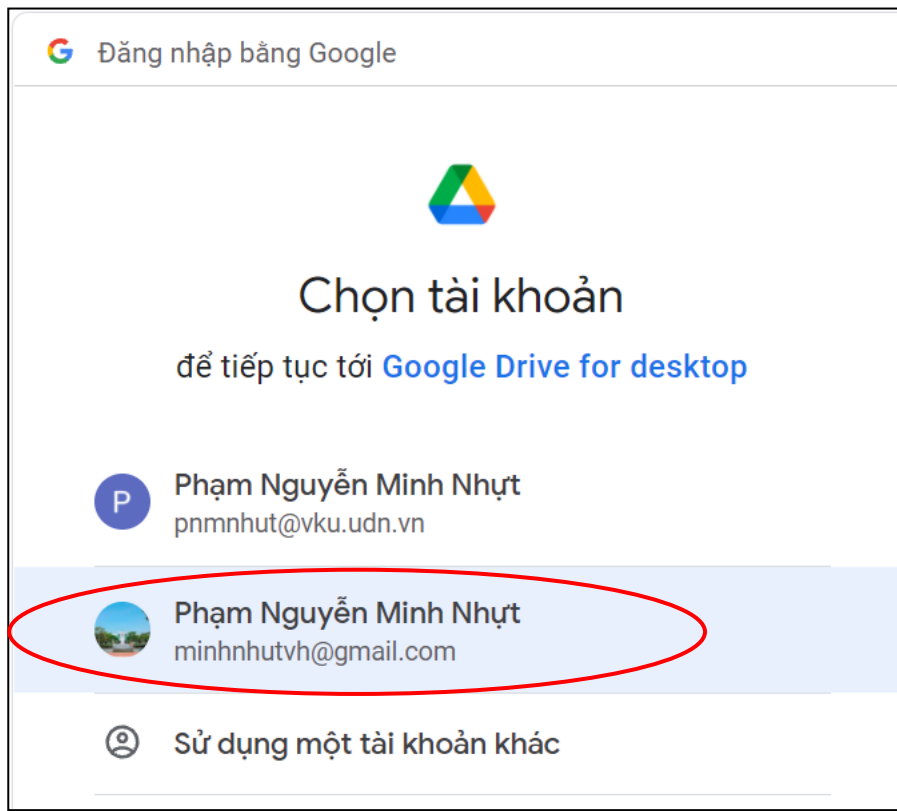
... RAM Ổ đĩa

```
from google.colab import drive
drive.mount('/content/gdrive')
```

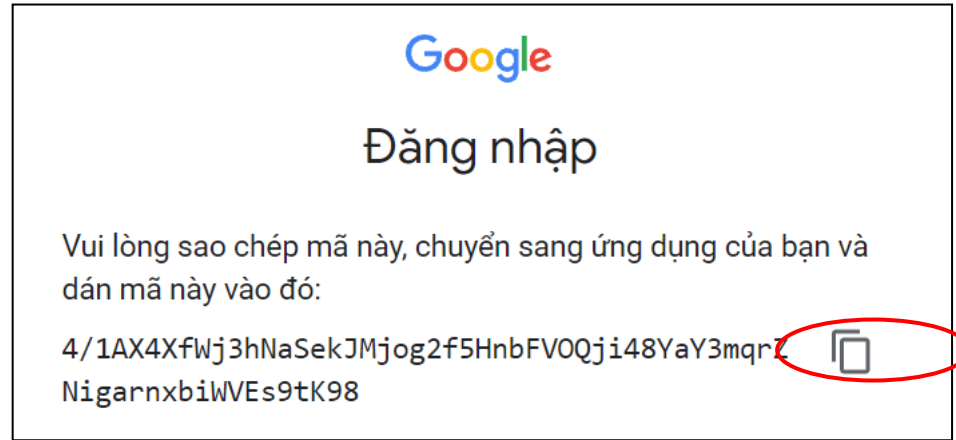
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com

Enter your authorization code:

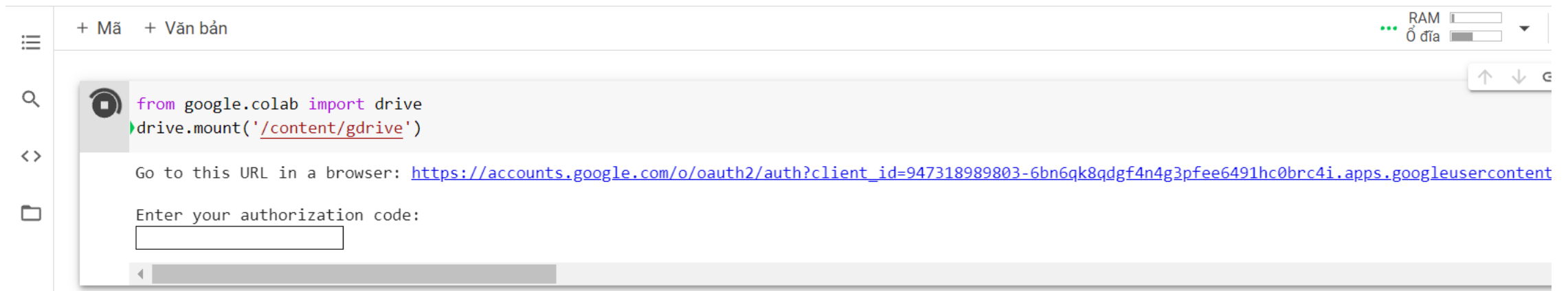
- Bước 8: Đăng nhập mail google



- Bước 9: Sao chép mã xác thực



- Bước 10: Dán mã vào ô Enter your authorization Code và nhấn enter



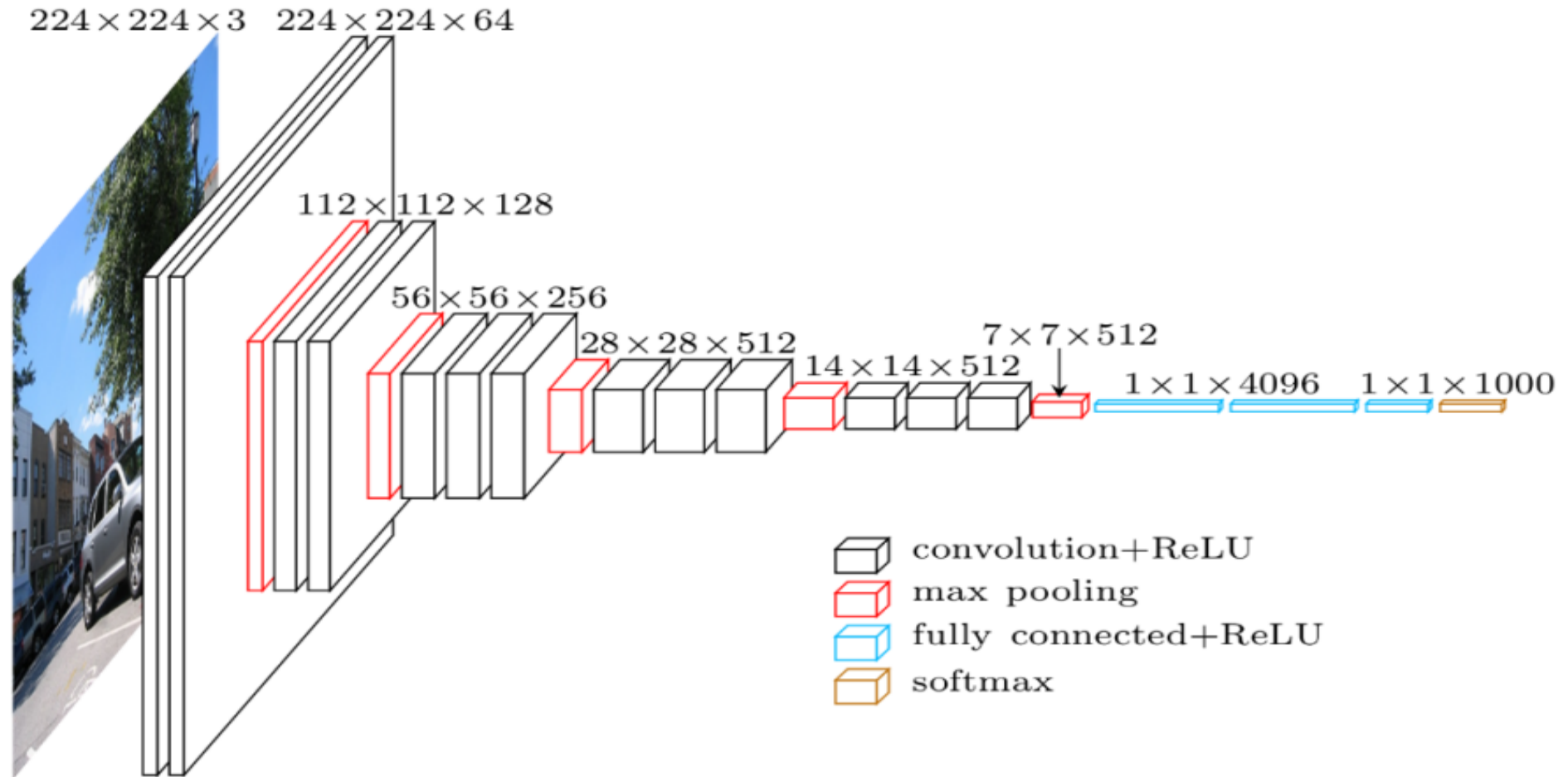
- Thêm mã: Nhấn vào + Mã
- Thực thi : Nhấn nút Tam giác



- Chuyển vào folder: `%cd "Path/folder"`
- Cài thêm thư viện: `!pip install <thư viện>`
- Chạy file python: `!python <file.py>`
- Chú ý: Nếu chạy file python có tham số thì thực hiện giống như tại terminal của máy cá nhân nhưng thêm dấu chấm thang (!) đầu lệnh

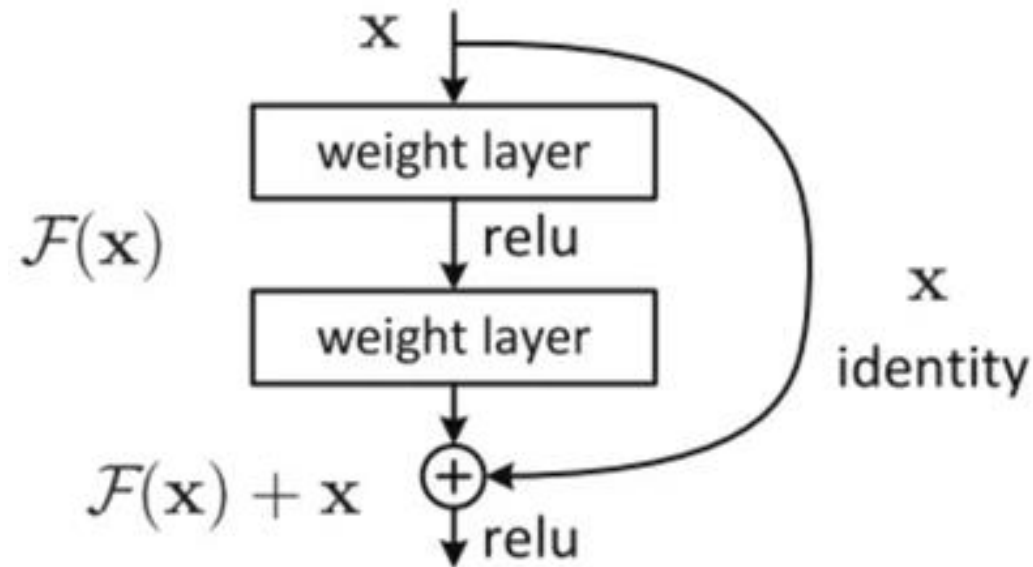
- Một số model pre-trained dùng cho phân lớp ảnh được tích hợp trong Keras
- VGGNet
 - Là mạng CNN sử dụng nhận dạng hình ảnh quy mô lớn
 - Được giới thiệu bởi Simonyan và Zisserman vào năm 2014
 - VGGNet có nhiều phiên bản
 - VGGNet16 → sử dụng 16 layer
 - VGGNet19 → sử dụng 19 layer
 - ...

- VGGNet

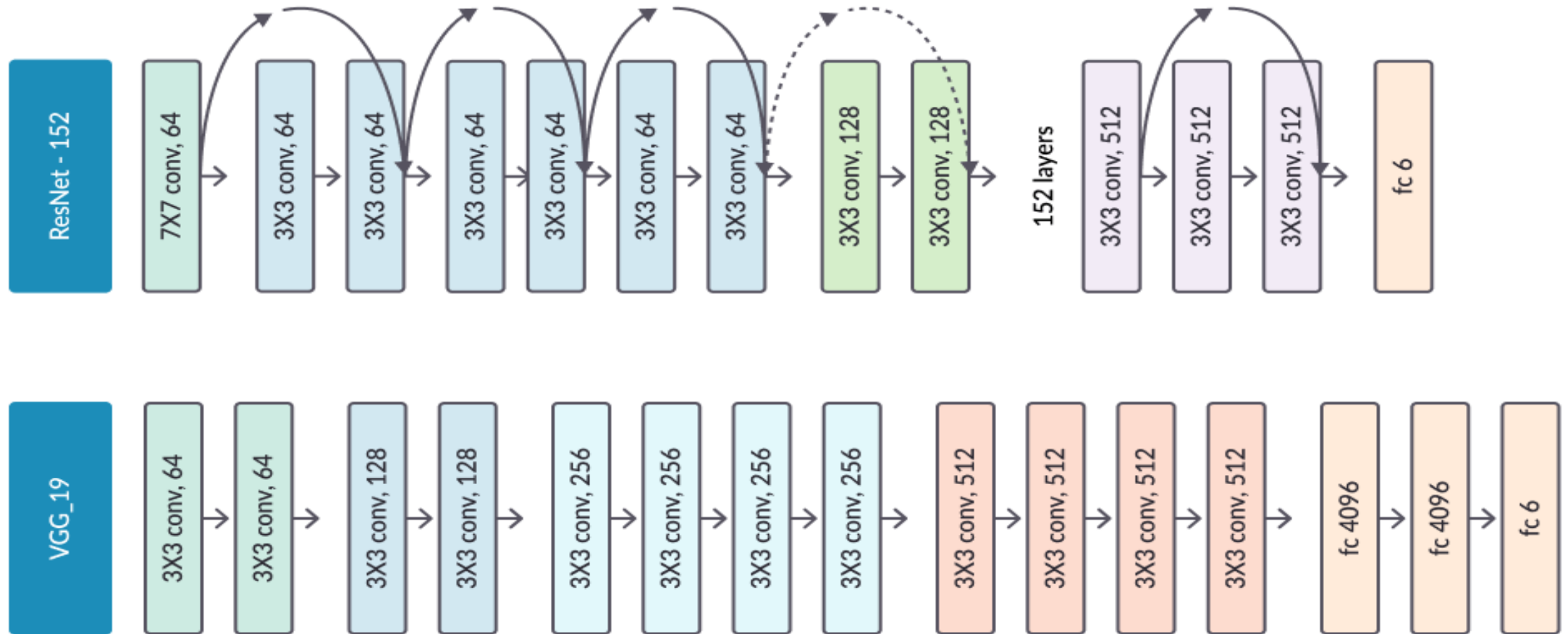


- ResNet50

- Công bố vào năm 2015 bởi He và các cộng sự → Được cập nhật vào năm 2016
- Kiến trúc mạng có 50 lớp
- Sử dụng kỹ thuật Residual Block → Kết nối tắt giữa các lớp

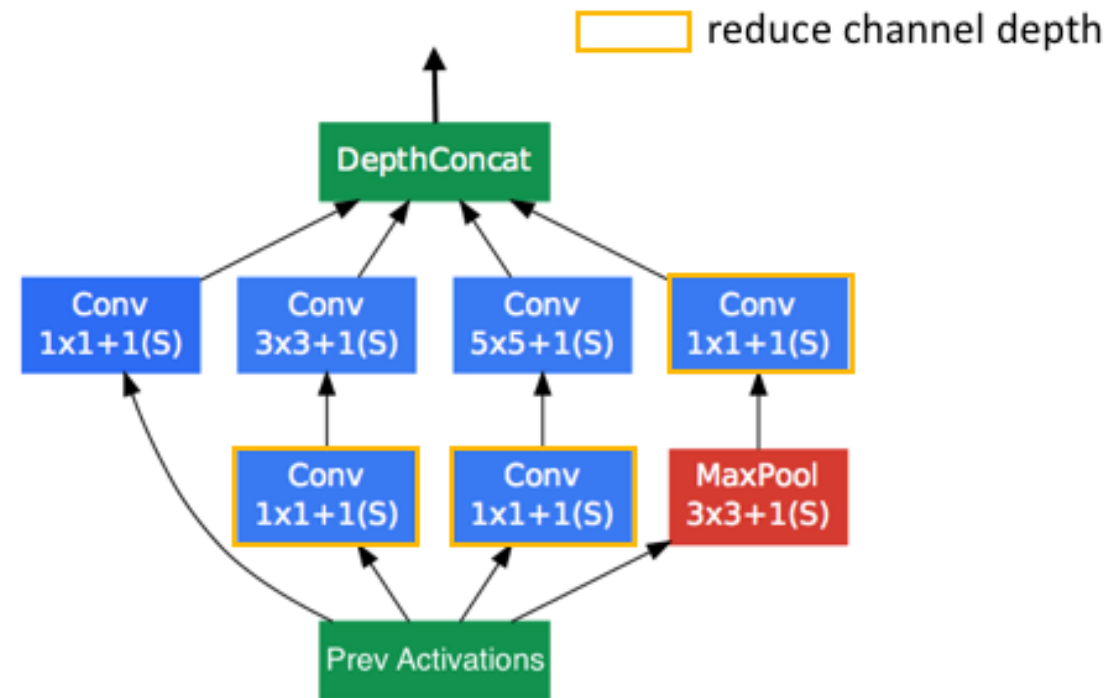


- ResNet50

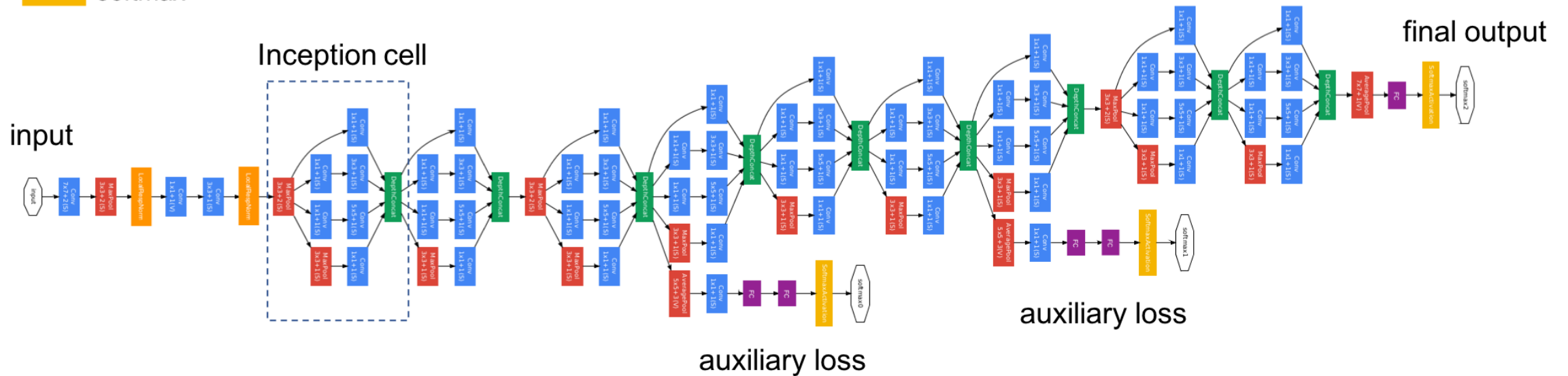
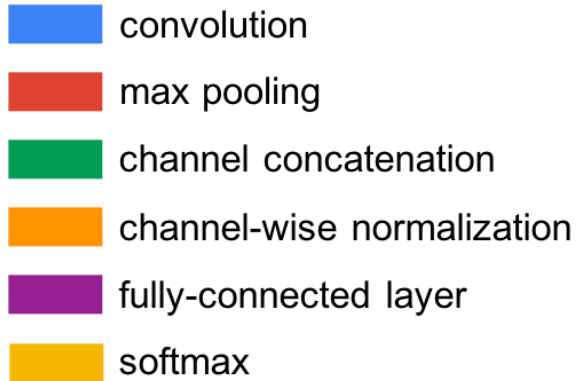


• Inception V3

- Được giới thiệu bởi Szegedy và các cộng sự vào năm 2014
- Sử dụng mô đun Inception
- → Giảm chiều của kênh
- → Rút trích đặc trưng multi-level



• Inception V3



- Xception
 - Được giới thiệu bởi François Chollet vào năm 2016
 - Xception là một phần mở rộng của kiến trúc Inception
 - Sử dụng kỹ thuật depthwise separable convolutions → tách lớp tích chập 2D thành tích chập 1D

- Tập dữ liệu ImageNet
 - Do Fei-Fei Li công bố vào năm 2009 Hội nghị về Thị giác Máy tính và Nhận dạng Mẫu (CVPR) ở Florida
 - ImageNet chứa hơn 20 nghìn danh mục
 - Hơn 14 triệu các hình ảnh đã được gán nhãn
 - Và được dùng để train các model được tích hợp trong Keras
 - <https://image-net.org/challenges/LSVRC/2014/index#data>

- Sử dụng các thư viện trong Keras

```
from keras.applications.resnet import ResNet50
from keras.applications.inception_v3 import InceptionV3
from keras.applications.xception import Xception
from keras.applications.vgg16 import VGG16
from keras.applications.vgg19 import VGG19
```

- Định nghĩa từ điển chứa các model

```
MODELS = { "vgg16": VGG16,
           "vgg19": VGG19,
           "inception": InceptionV3,
           "xception": Xception,
           "resnet": ResNet50 31 }
```

- Sử dụng các thư viện trong Keras: Có 2 cách

- Cách 1:

- Lưu model về thành file *.hdf5
 - Sau đó Load model để dùng

```
from keras.applications.vgg19 import VGG19

model = VGG19()
model.save("modelvgg19.hdf5")
model.summary() # Hiển thị tóm tắt các tham số của model
```

- Cách 2: Load model sử dụng trực tiếp

- Load model về bộ nhớ và thực hiện dự đoán phân lớp ảnh (diễn ra cùng thời điểm)
- Thực hành sử dụng các model pre-trained đã tích hợp trong Keras