

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HCM**

**NGÀNH KHOA HỌC MÁY TÍNH**

-----🙠🙟🕮🙝🙢------

**BÁO CÁO ĐỒ ÁN**

**MÔN: ĐA PHƯƠNG TIỆN**

**ĐỀ TÀI: NHẬN DIỆN GIỐNG CHÓ**

**GVHD: MAI TIẾN DŨNG**

**SINH VIÊN: LÊ CÔNG MINH**

**LỚP: CS232.K21**

**MSSV: 18521099**

TP. HỒ CHÍ MINH, THÁNG 6 NĂM 2020

**GIỚI THIỆU**

Bài toán nhận diện giống với mục địch xác định giống chó thông qua hình ảnh

Input: Gồm các ảnh của các loại chó và các nhãn tương ứng

Output: Xác định giống chó thông qua hình ảnh

Bài toán sử dụng các pretrained model của mạng CNN, cụ thể là VGG16, Xception

Ứng dụng: tìm chó lạc, phân loại chó để định giá, …

Source code: <https://github.com/18521099/DAPHUONGTIEN>

**THU NHẬP VÀ XỬ LÍ DATASET**

Dataset được thu nhập từ trang Kaggle: <https://www.kaggle.com/c/dog-breed-identification>

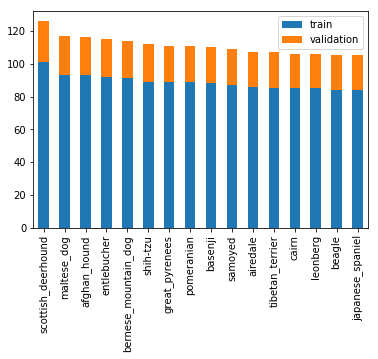
Dataset gồm các file:

* Train: Gồm hơn 10000 ảnh của các loại chó
* Test: Gồm hơn 10000 ảnh của các loại chó
* Labels: Gồm các ID và nhãn tương ứng của các loại chó trong file Train, tổng cộng 120 giống chó

**Một số hình ảnh dataset**



Vì dataset nhiều nên việc train khá lâu, trong quá trình train mà gặp vấn đề sẽ mất nhiều thời gian nên em quyết định giảm số lớp từ 120 => 16 lớp chó xuất hiện nhiều nhất, gồm 1777 tấm hình

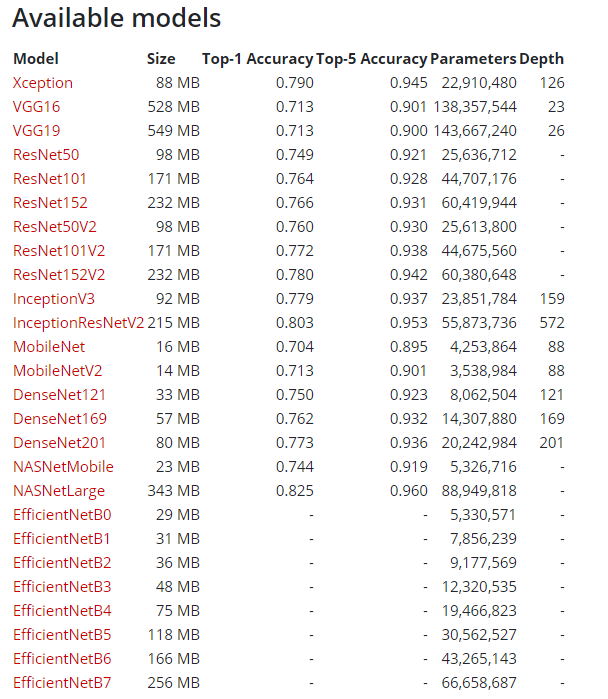


Cột xanh là số lượng ảnh dùng để train, còn lại dùng để test, nhìn vào dữ liệu ta thấy 16 lớp này có dữ liệu khá cân bằng, rất thuận tiện cho việc dự đoán

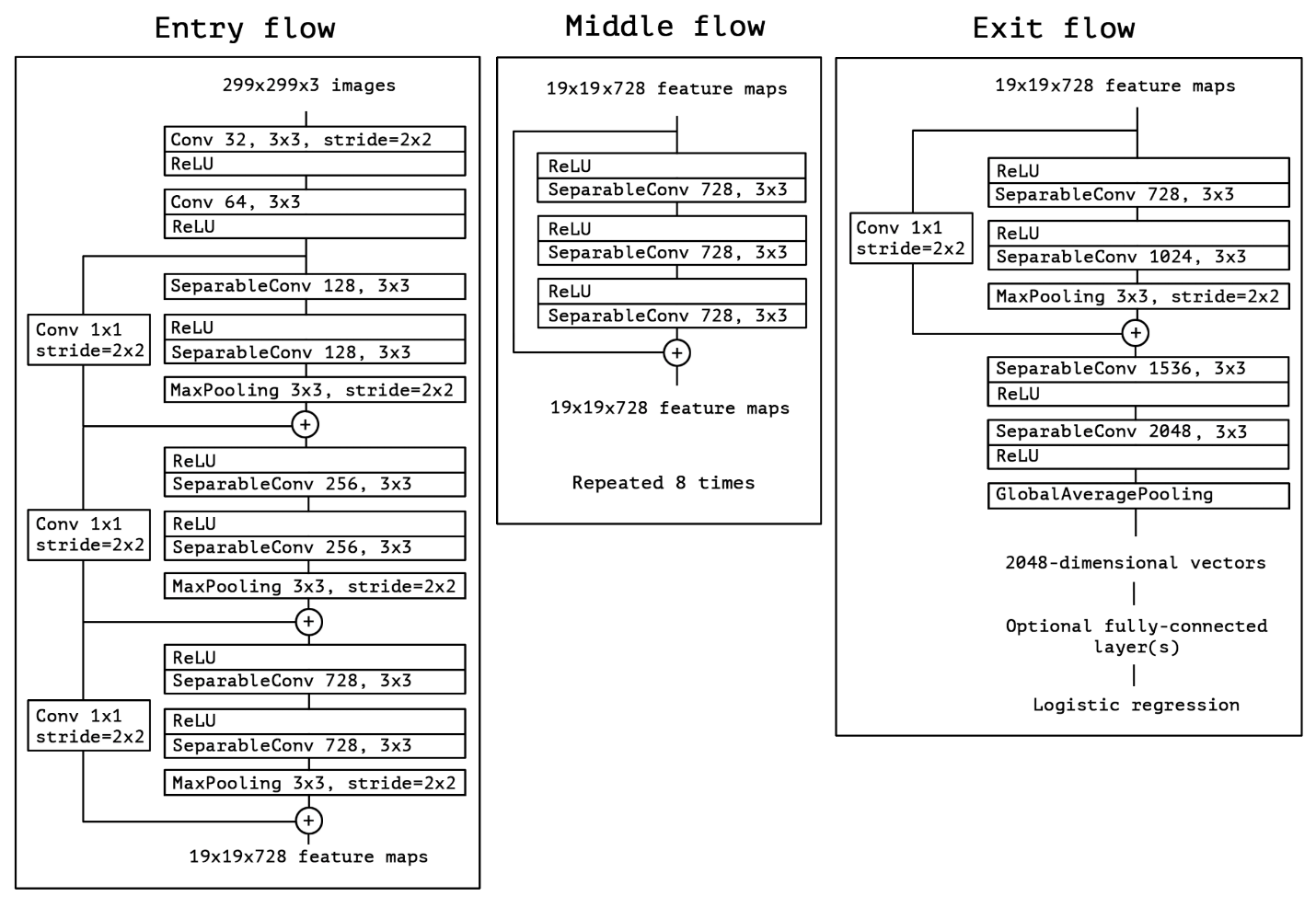
**TỔNG QUAN VGG16, XCEPTION**

VGG16, Xception là mạng convolutional neural network, model sau khi train bởi mạng VGG16, Xception đạt độ chính xác hơn 90% trong top-5 test trong dữ liệu ImageNet gồm 14 triệu ảnh thuộc 1000 lớp khác nhau

**Các model được thống kê bởi keras**



**Kiến trúc Xception**



**Kiến trúc VGG16**

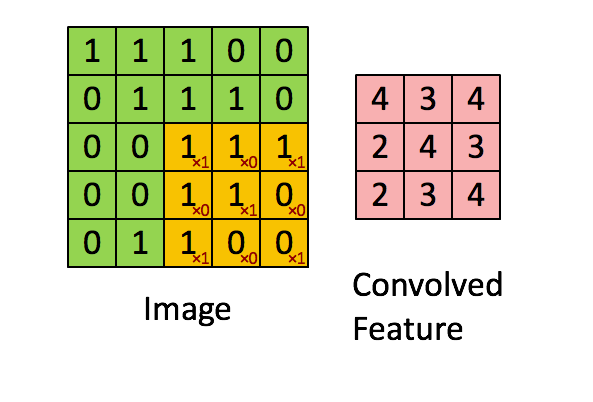
****

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 1 layer 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.

Khi sử dụng các pretrained model này có 2 params cần chú ý:

* include\_top (True/False): có sử dụng Fully Connected hay không
* weights (‘imagenet’,None): có sử dụng pretrained weights của imagenet hay không

**Convolutional layer**



Các convolutional layer có các parameter(kernel) đã được học để tự điều chỉnh lấy ra những thông tin chính xác nhất mà không cần chọn các feature.

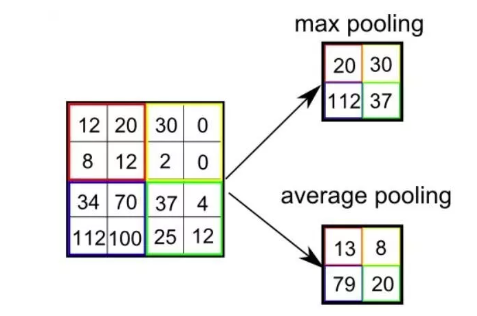
Trong hình ảnh ví dụ trên, ma trận bên trái là một hình ảnh trắng đen được số hóa. Ma trận có kích thước 5×5 và mỗi điểm ảnh có giá trị 1 hoặc 0 là giao điểm của dòng và cột.

Convolution hay tích chập là nhân từng phần tử trong ma trận 3. Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ như trong ví dụ trên là 3×3.

**Pooling layer**

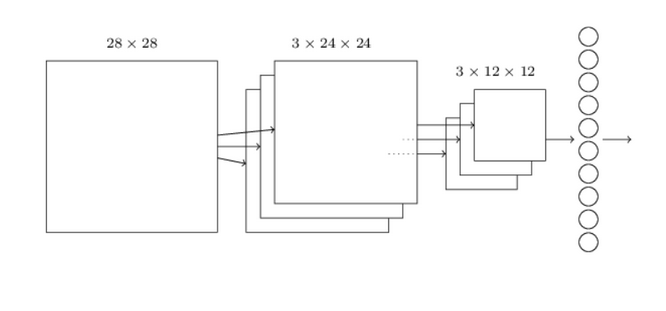
Lớp pooling thường được sử dụng ngay sau lớp convulational để đơn giản hóa thông tin đầu ra để giảm bớt số lượng neuron.

Có 2 loại pooling phổ biến là max-pooling, avg-pooling, các thủ tục này chọn giá trị lớn hay trung bình cộng vùng đầu vào.

****

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.

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.



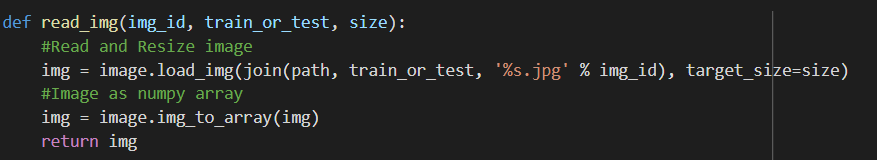
2 lớp cuối cùng của các kết nối trong mạng là một lớp đầy đủ kết nối (fully connected layer) . Lớp này nối mọi nơron từ pooling layer tới mọi nơron của tầng ra.

**XỬ LÍ DỮ LIỆU**

Các dataset có kích thước khác nhau nên ta phải resize các ảnh về cùng kích thước theo yêu cầu của mỗi model

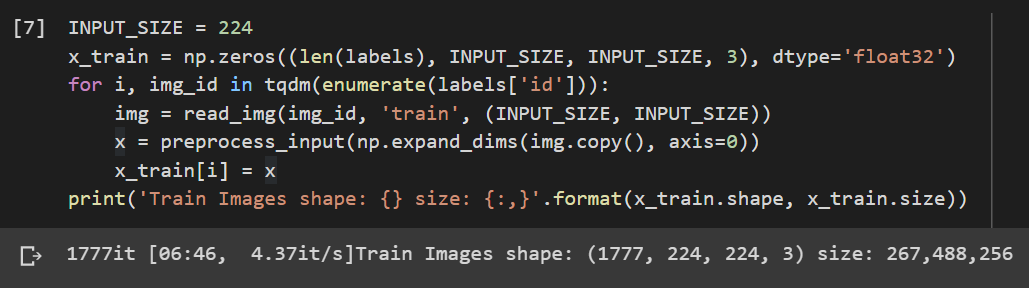
Sau khi resize, ta chuyển ảnh về dạng mảng

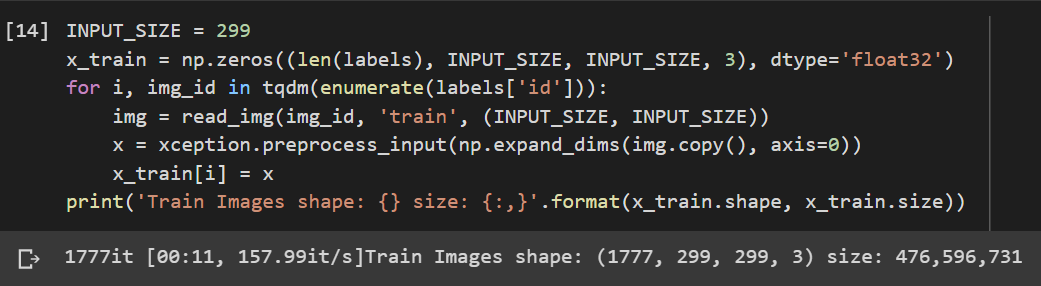
Hàm dùng để resize và chuyển ảnh về dạng mảng



Mỗi model yêu cầu giá trị trong từng phần tử của mảng nằm trong một khoảng qui định, nên ta phải dùng phương thức preprocess\_input để chuyển mảng sang dạng model yêu cầu.

**Xử lí ảnh theo model VGG16**



**Xử lí ảnh theo Xception** 

**Ảnh trước khi preprocess**



**Ảnh sau khi preprocess**

**VGG16 Xception**

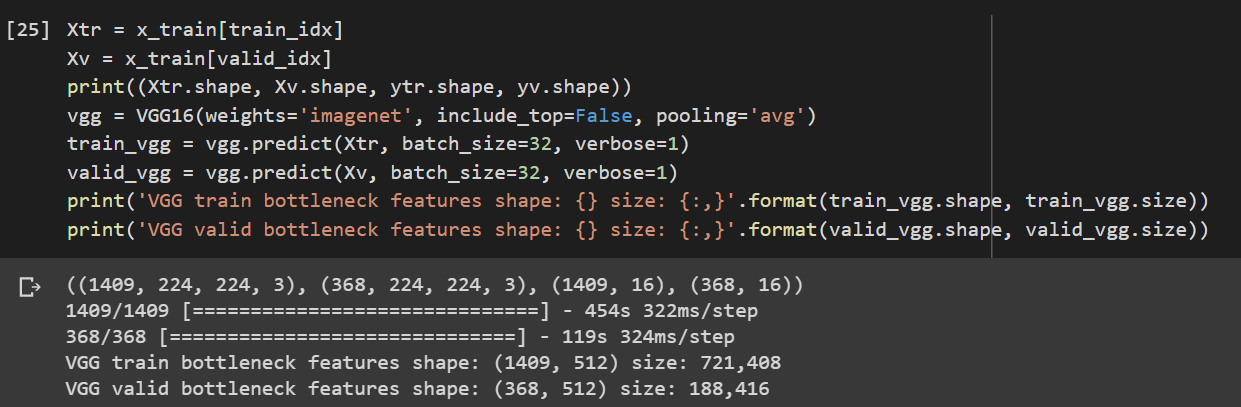
****

Ảnh sau khi preprocess của Xception khá tối còn VGG16 thì ngược lại

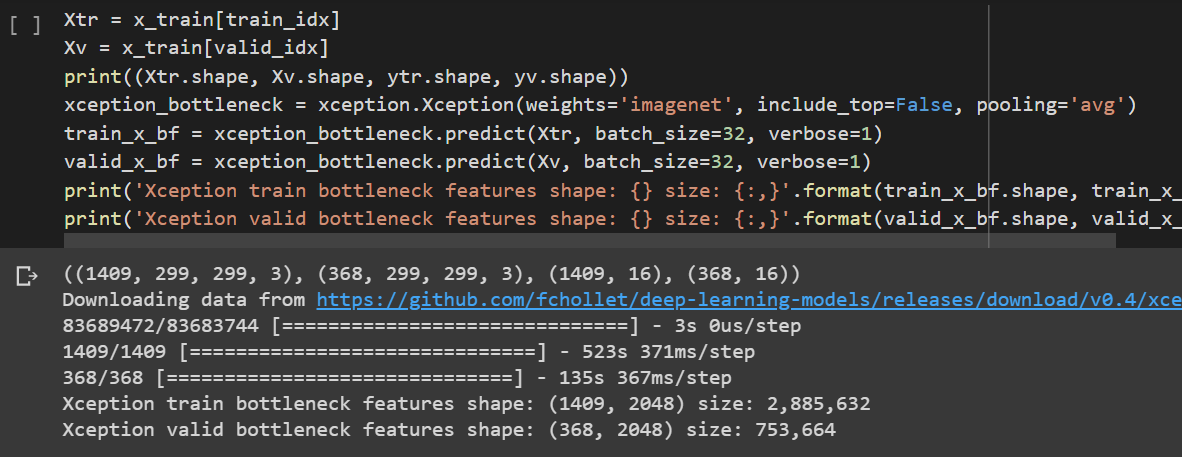
**TRAINING AND TESTING**

Sau khi xử lí ảnh sau, ta gọi model và train

**VGG16**

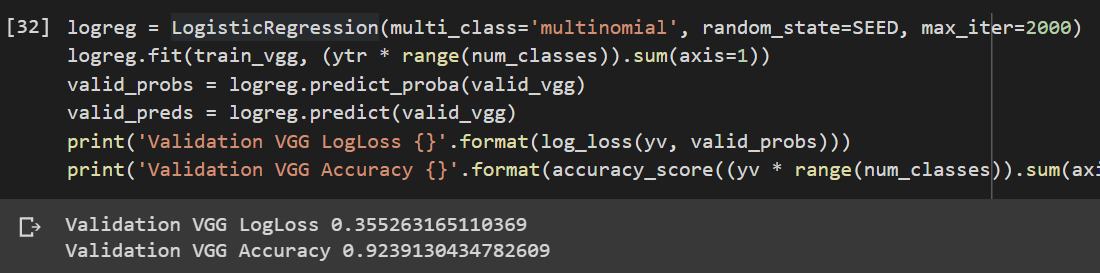


**Xception**

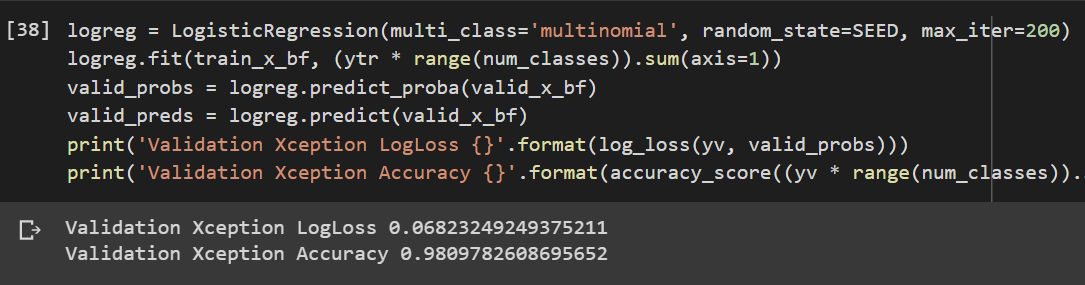


Đối với 2 model trên, ta sử dụng là các weights của imagenet, sử dụng average pooling, và bỏ phần fully connected layer thì model sẽ lấy ra các đặc điểm của ảnh như tai, mắt, lông,… rồi ta dùng các đặc điểm này làm input cho các bài toán thông thường như linear regression, logistic regression,…

Training VGG16



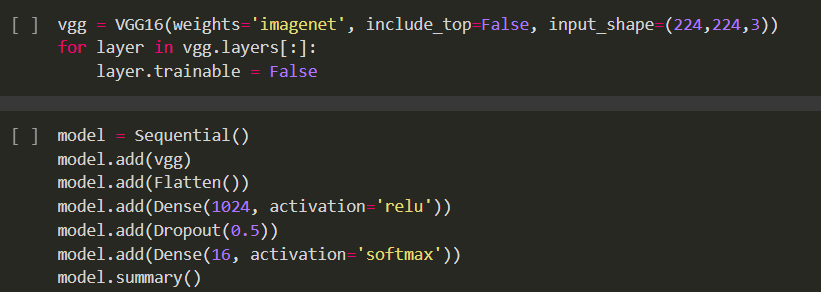
Training Xception

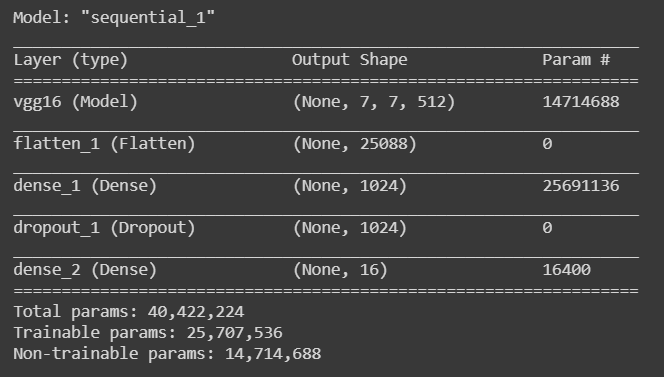


Ta thấy accuracy của VGG16 nhỏ hơn Xception, LogLoss của VGG16 lớn hơn Xception

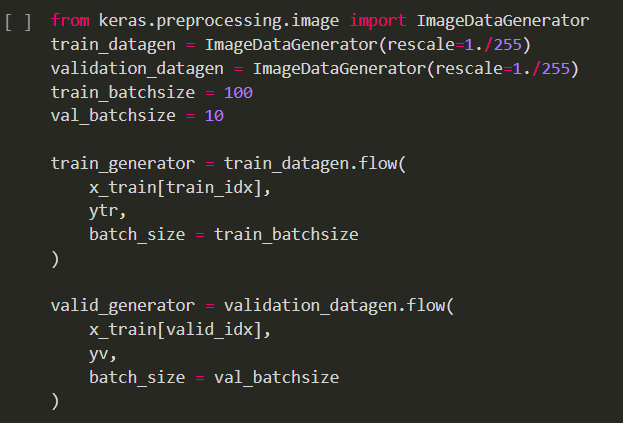
**FINE TUNING**

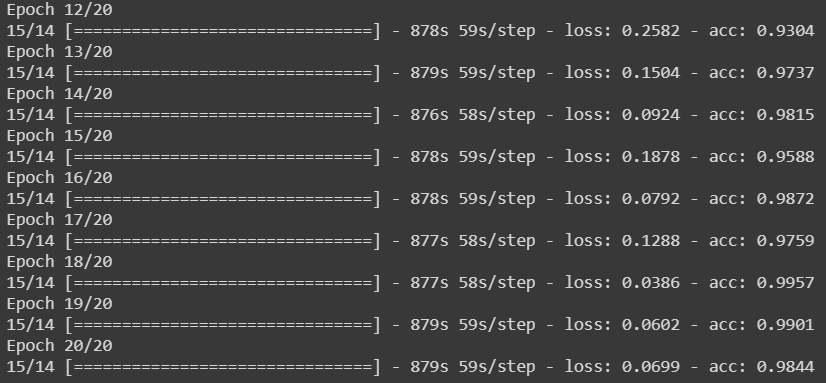
Bỏ đi các FC layers thêm vào các lớp mới





Dùng ImageDataGenerator tạo ra them nhiều ảnh hơn





Sau khi train accuracy lên tới 98%

**TÀI LIỆU THAM KHẢO**

<https://keras.io/api/applications/>

<https://www.kaggle.com/c/dog-breed-identification/notebooks>

<https://nttuan8.com/bai-6-convolutional-neural-network/#Mang_VGG_16>

<https://nttuan8.com/bai-9-transfer-learning-va-data-augmentation/>

<https://viblo.asia/p/gioi-thieu-ve-cac-pre-trained-models-trong-linh-vuc-computer-vision-3Q75wB1GlWb>