HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BÁO CÁO ĐỀ TÀI

XÂY DỰNG HỆ THỐNG PHÂN LOẠI HOA QUẢ

Contents

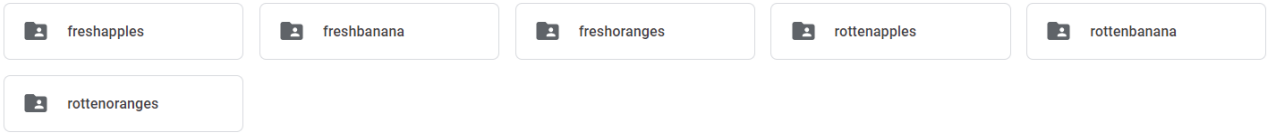
I , Giới thiệu tập dữ liệu FRUIT 3
II ,Giới thiệu về CNN 4
1.Convolutional layer 4
2.Hàm Relu 5
3.Pooling layer 6
4.Fully connected layer 7
5.Tóm tắt 7
III ,Các bước giải bài toán 8
1.1. Tải xuống và đọc tập dữ liệu 8
1.2. Chuẩn hóa hình ảnh(Image normalize) 10
2 .Xây dựng mạng CNN cho model phân loại hoa quả và độ tươi héo của hoa quả10
2.1 . Một số khái niệm liên quan 10
2.1.1 Keras là gì 10
2.1.2 Sequential model 10
2.1.3 Tạo Convolutionnal Layers 11
2.1.4 Pooling Layers: 11
2.1.5 Dense () 11
2.1.6 Hàm compile: 11
2.1.7 Hàm fit (): 11
2.1.8 Dropout: 12
2.1.9 Flatten 12
2.2 . Giới thiệu về mô hình MobinetV2 13
2.2.1 MobileNet 13
2.2.2 MobileNetV2: 15
Áp dụng trong bài toán nhận diện màu sắc hoa quả: 17
2.3 Xây dựng mô hình 18
Code: 18
Bản tóm tắt : 19
2.4 Quy trình đào tạo 20
2.5 Hình dung quá trình đào tạo 20

2.6 Đánh giá mô hình 21
3. Lưu lịch sử đào tạo và mô hình 21
3.1 Lưu lịch sử đào tạo 21
3.2 Lưu mô hình 21
4. Xây dựng mô hình phân loại độ xanh chín của hoa quả bằng thuật toán KMEAN 22
4.1. Giới thiệu về thuật toán KMEAN và các ứng dụng của nó 22
4.2. Xây dựng thuật toán tìm màu gần nhất 26
5. Đẩy mô hình lên web và thử nghiệm: 27
5.1. Giới thiệu sơ qua về Flask: 27
5.1.1. Python Flask là gì? 27
5.1.2.Tính năng của Flask Framework 30
Tại sao sử dụng Flask? 30
5.2 Hướng dẫn chạy project 31
5.3 . Test thử nghiệm 33
Test thử 1 ảnh thứ nhất : 33
Test thử ảnh thứ 2 : 34
Test thử ảnh thứ 3: 34

I , Giới thiệu tập dữ liệu FRUIT

Dữ liệu được chia làm 2 tập train và test

Mỗi tập train và test bao gồm dữ liệu của

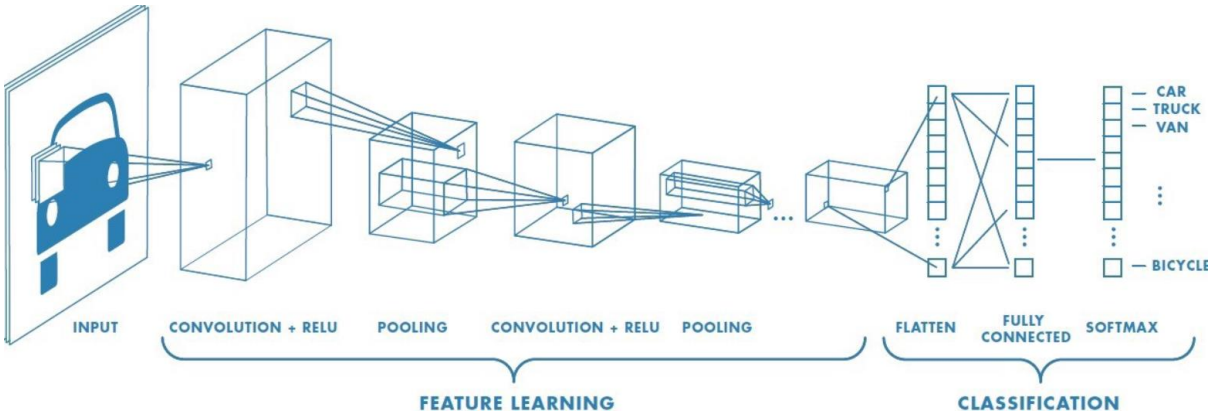


Freshapples: Chứa ảnh táo xanh

Freshbanana: Chứa ảnh chuối xanh
Freshoranges: Chứa ảnh cam xanh
Rottenapples: Chứa ảnh táo hỏng

Rottenbanana: Chứa ảnh chuối hỏng
Rottenoranges: Chứa ảnh cam hỏng

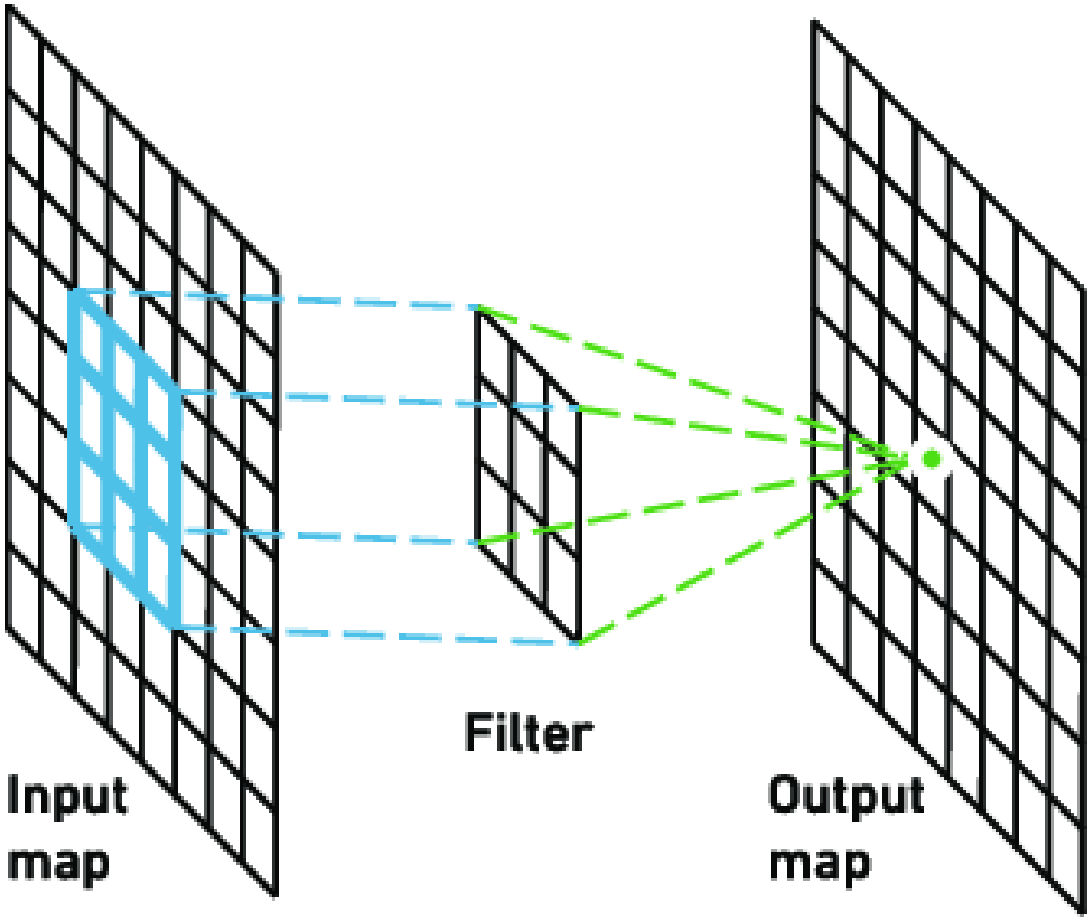
II ,Giới thiệu về CNN



Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) 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. 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:

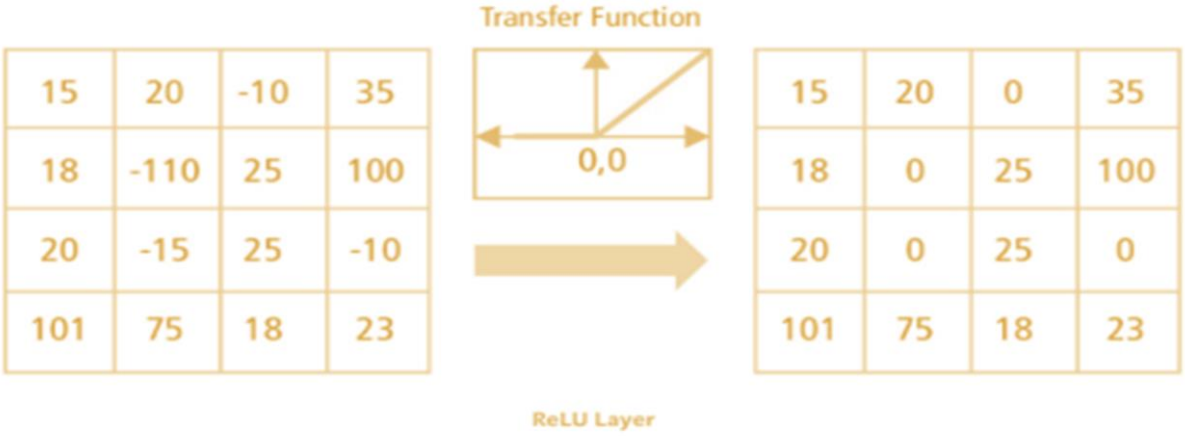
1.Convolutional layer

Đây là lớp quan trọng nhất của CNN, lớp này có nhiệm vụ thực hiện mọi tính toán. Nếu lớp này có số lượng nhiều thì chương trình chạy càng được cải thiện. Sử dụng các layer với 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 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.



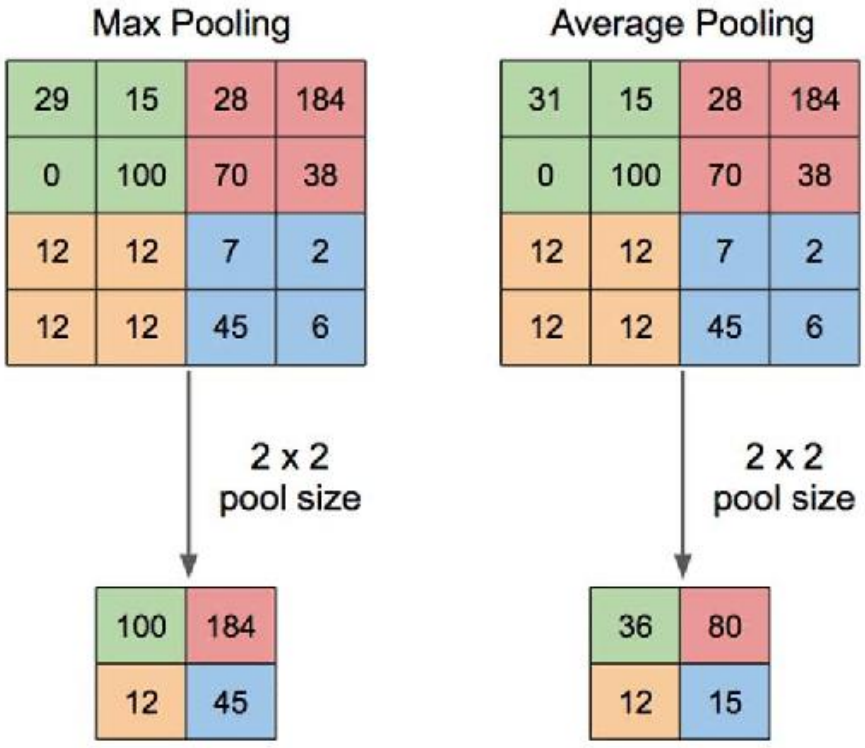
2.Hàm Relu

Relu layer (Rectified Linear Unit) là hàm kích hoạt trong neural network và hàm này còn được gọi là activation function.ReLU với đầu ra là: $f(x) = \max(0, x)$ thì đây là một hàm kích hoạt có tác dụng mô phỏng các neuron có tỷ lệ truyền xung qua axon. Trong activation function thì nó còn có hàm nghĩa là: Relu, Leaky, Tanh, Sigmoid, Maxout,...Hiện nay, hàm relu được dùng phổ biến và vô cùng thông dụng.
Nó được sử dụng nhiều cho các nhu cầu huấn luyện mạng neuron thì relu mang lại rất nhiều ưu điểm nổi bật như: việc tính toán sẽ trở nên nhanh hơn,... Quá trình sử dụng relu, chúng ta cần lưu ý đến vấn đề tùy chỉnh các learning rate và theo dõi dead unit. Những lớp relu layer đã được sử dụng sau khi filter map được tính ra và áp dụng hàm relu lên những giá trị của filter map.



3.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:



Nguyên lý hoạt động 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

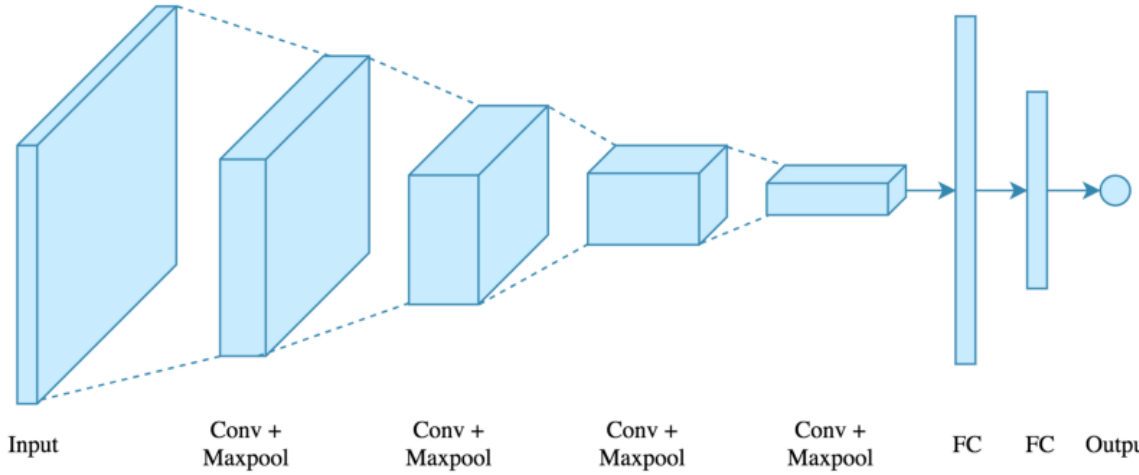
pooling window có kích thước là 2 * 2, stride = 2 để đảm bảo không trùng nhau, và áp dụng max pooling / average pooling.

4.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 cấu trúc một 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.

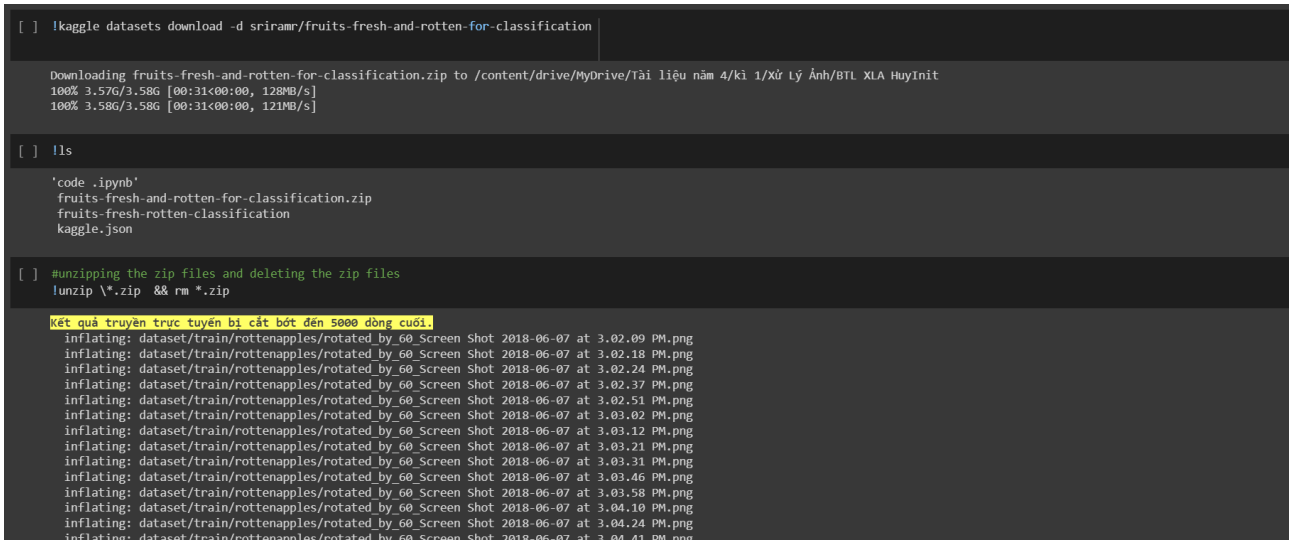
5.Tóm tắt

-Đầu vào của lớp tích chập là hình ảnh

-Chọn đối số, áp dụng các bộ lọc với các bước nhảy, padding nếu cần. Thực hiện tích chập cho hình ảnh và áp dụng hàm kích hoạt ReLU cho ma trận hình ảnh.
-Thực hiện Pooling để giảm kích thước cho hình ảnh.
-Thêm nhiều lớp tích chập sao cho phù hợp
-Xây dựng đầu ra và dữ liệu đầu vào thành 1 lớp được kết nối đầy đủ (Full Connected)
-Sử dụng hàm kích hoạt để tìm đối số phù hợp và phân loại hình ảnh.

III ,Các bước giải bài toán

1.1. Tải xuống và đọc tập dữ liệu



```
[ ]  import glob

     train_fresh_apple_dir = './dataset/train/freshapples'
     train_rotten_apple_dir = './dataset/train/rottenapples'
     train_fresh_banana_dir = './dataset/train/freshbanana'
     train_rotten_banana_dir = './dataset/train/rottenbanana'
     train_fresh_orange_dir = './dataset/train/freshoranges'
     train_rotten_orange_dir = './dataset/train/rottenoranges'

     test_fresh_apple_dir = './dataset/test/freshapples'
     test_rotten_apple_dir = './dataset/test/rottenapples'
     test_fresh_banana_dir = './dataset/test/freshbanana'
     test_rotten_banana_dir = './dataset/test/rottenbanana'
     test_fresh_orange_dir = './dataset/test/freshoranges'
     test_rotten_orange_dir = './dataset/test/rottenoranges'

     train_fresh_apple_files = glob.glob(train_fresh_apple_dir + '/*')
     train_rotten_apple_files = glob.glob(train_rotten_apple_dir + '/*')
     train_fresh_banana_files = glob.glob(train_fresh_banana_dir + '/*')
     train_rotten_banana_files = glob.glob(train_rotten_banana_dir + '/*')
     train_fresh_orange_files = glob.glob(train_fresh_orange_dir + '/*')
     train_rotten_orange_files = glob.glob(train_rotten_orange_dir + '/*')

     print('train samples of fresh apple:', len(train_fresh_apple_files))
     print('train samples of rotten apple:', len(train_rotten_apple_files))
     print('train samples of apple:', len(train_fresh_apple_files) + len(train_rotten_apple_files))
     print('train samples of fresh banana:', len(train_fresh_banana_files))
     print('train samples of rotten banana:', len(train_rotten_banana_files))
     print('train samples of banana:', len(train_fresh_banana_files) + len(train_rotten_banana_files))
     print('train samples of fresh orange:', len(train_fresh_orange_files))
     print('train samples of rotten orange:', len(train_rotten_orange_files))
     print('train samples of orange:', len(train_fresh_orange_files) + len(train_rotten_orange_files))
     print('total train samples:',
           len(train_fresh_apple_files) +
           len(train_rotten_apple_files) +
           len(train_fresh_banana_files) +
```

```
train samples of fresh apple: 1693
train samples of rotten apple: 2342
train samples of apple: 4035
train samples of fresh banana: 1581
train samples of rotten banana: 2224
train samples of banana: 3805
train samples of fresh orange: 1466
train samples of rotten orange: 1595
train samples of orange: 3061
total train samples: 10901
*****************************
test samples of fresh apple: 395
test samples of rotten apple: 601
test samples of apple: 996
test samples of fresh banana: 381
test samples of rotten banana: 530
test samples of banana: 911
test samples of fresh orange: 388
test samples of rotten orange: 403
test samples of orange: 791
total test samples: 2698
```

1.2. Chuẩn hóa hình ảnh(Image normalize)

```
[ ]  import os
     from tensorflow.keras.layers import Conv2D
     from tensorflow.keras.layers import GlobalMaxPool2D
     from tensorflow.keras.layers import Flatten
     from tensorflow.keras.layers import Dense
     from tensorflow.keras.layers import Dropout
     from tensorflow.keras.layers import InputLayer
     from tensorflow.keras.layers import GlobalAveragePooling2D
     from tensorflow.keras.models import Sequential
     from tensorflow.keras import optimizers
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.regularizers import l2

     train_datagen = ImageDataGenerator(rescale=1./255,
                                        zoom_range=0.3,
                                        rotation_range=50,
                                        width_shift_range=0.2,
                                        height_shift_range=0.2,
                                        shear_range=0.2,
                                        horizontal_flip=True,
                                        fill_mode='nearest')

     val_datagen = ImageDataGenerator(rescale=1./255)
     dataset_path = './dataset' #/content/drive/MyDrive/Tài liệu năm 4/kì 1/Xử Lý Ảnh/BTL XLA HuyInit/dataset/
     train_set_path = os.path.join(dataset_path, 'train')
     val_set_path = os.path.join(dataset_path, 'test')

     BATCH_SIZE = 64
     TARGET_SIZE = input_shape[:2]
     train_generator = train_datagen.flow_from_directory(train_set_path,
                                                         target_size = TARGET_SIZE,
                                                         batch_size = BATCH_SIZE,
                                                         class_mode = 'categorical')

     val_generator = val_datagen.flow_from_directory(val_set_path,
                                                     target_size = TARGET_SIZE,
```

2 .Xây dựng mạng CNN cho model phân loại hoa quả và độ tươi héo của hoa quả

2.1 . Một số khái niệm liên quan

2.1.1 Keras là gì

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. Với những người mới tiếp cận đến Deep như mình thì mình chọn sử dụng Keras để build model vì nó đơn giản,dễ nắm bắt hơn các thư viện khác. Dưới đây mình xin giới thiệu một chút về API này.

2.1.2 Sequential model

Trong Keras có hỗ trợ 2 cách dựng models là Sequential model và Function API .Kiểu mô hình Sequential cho phép bạn xây dựng mô hình theo từng lớp , tuần tự

2.1.3 Tạo Convolutionnal Layers

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. Đặc điểm mỗi hàm các bạn có thể search thêm để biết cụ thể nó ntn.
- padding : có thể là "valid" hoặc "same". Với same thì có nghĩa là padding =1.

2.1.4 Pooling Layers:

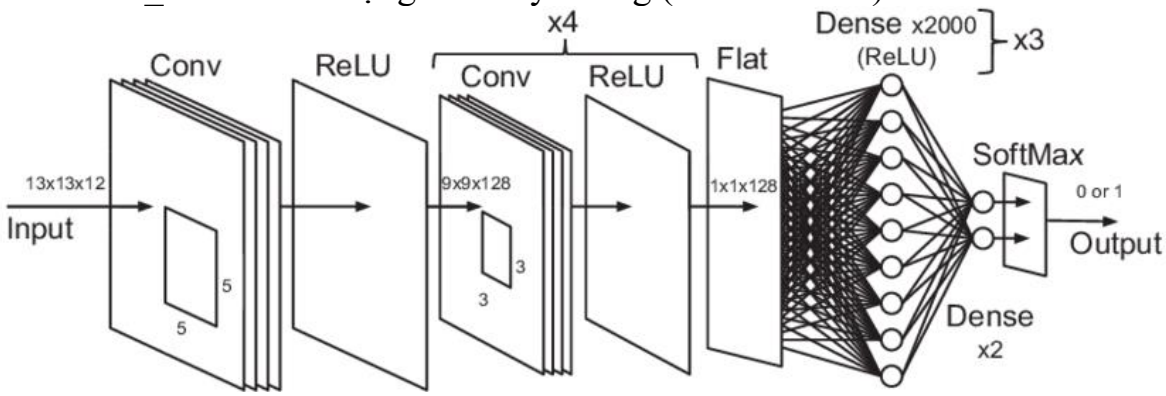
Được sử dụng để làm giảm param khi train, nhưng vẫn giữ được đặc trưng của ảnh.

- 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.

2.1.5 Dense ()

Layer này cũng như một layer neural network bình thường, với các tham số sau:

- units : số chiều output, như số class sau khi train (chó , mèo, lợn, gà).
- activation : chọn activation đơn giản với sigmoid thì output có 1 class.
- use_bias : có sử dụng bias hay không (True or False)



2.1.6 Hàm compile:

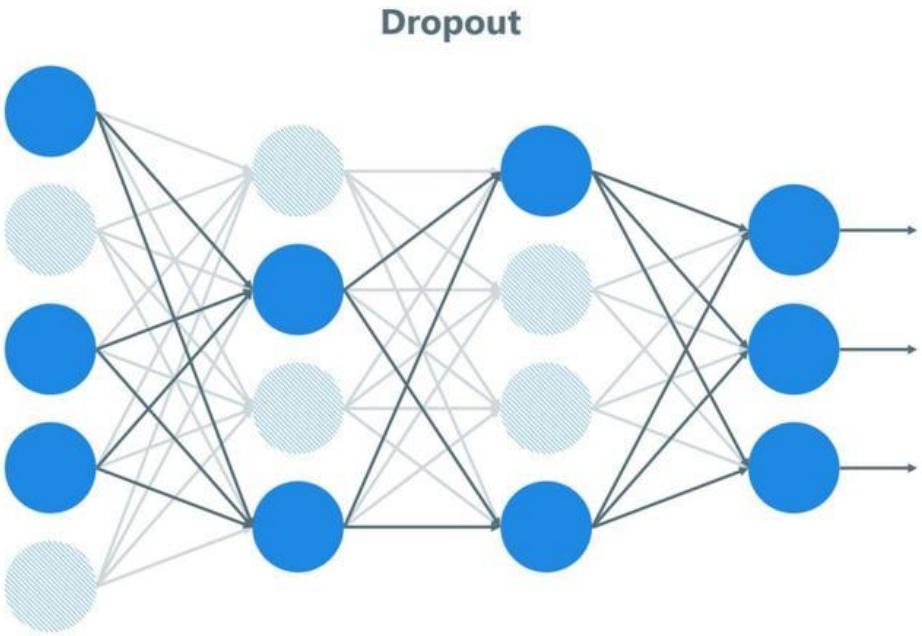
Ở hàm này chúng ta sử dụng để training models như thuật toán train qua optimizer như Adam, SGD, RMSprop,..
learning_rate : dạng float , tốc độc học, chọn phù hợp để hàm số hội tụ nhanh.

2.1.7 Hàm fit ():

-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.

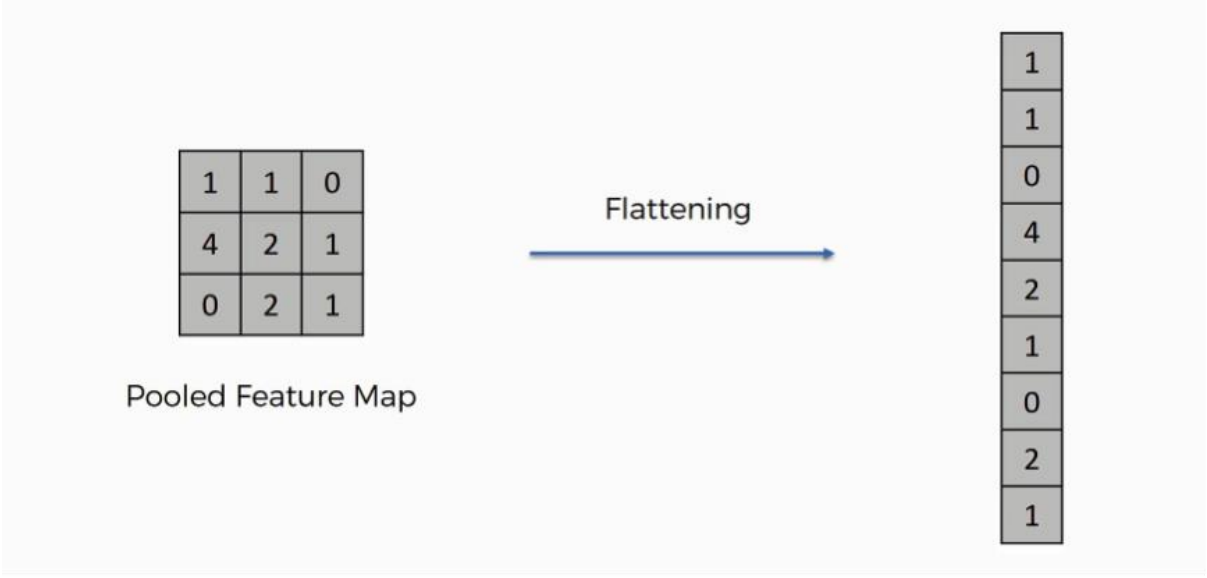
2.1.8 Dropout:

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).



2.1.9 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 mapthành vectơ một chiều là vì vectơ này bây giờ sẽ được đưa vào một mạng nơ-ron nhân tạo . 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.

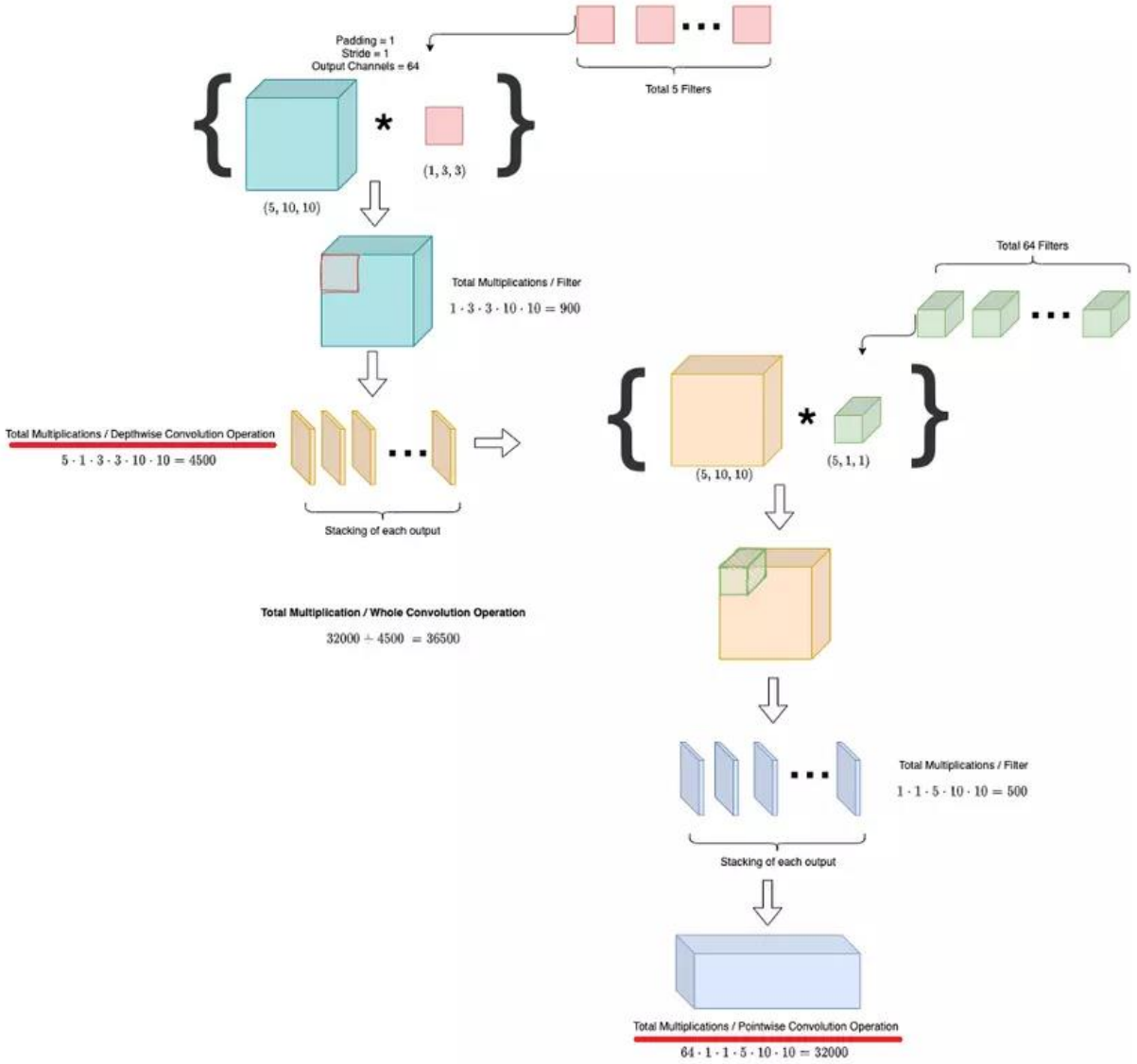


2.2 . Giới thiệu về mô hình MobinetV2

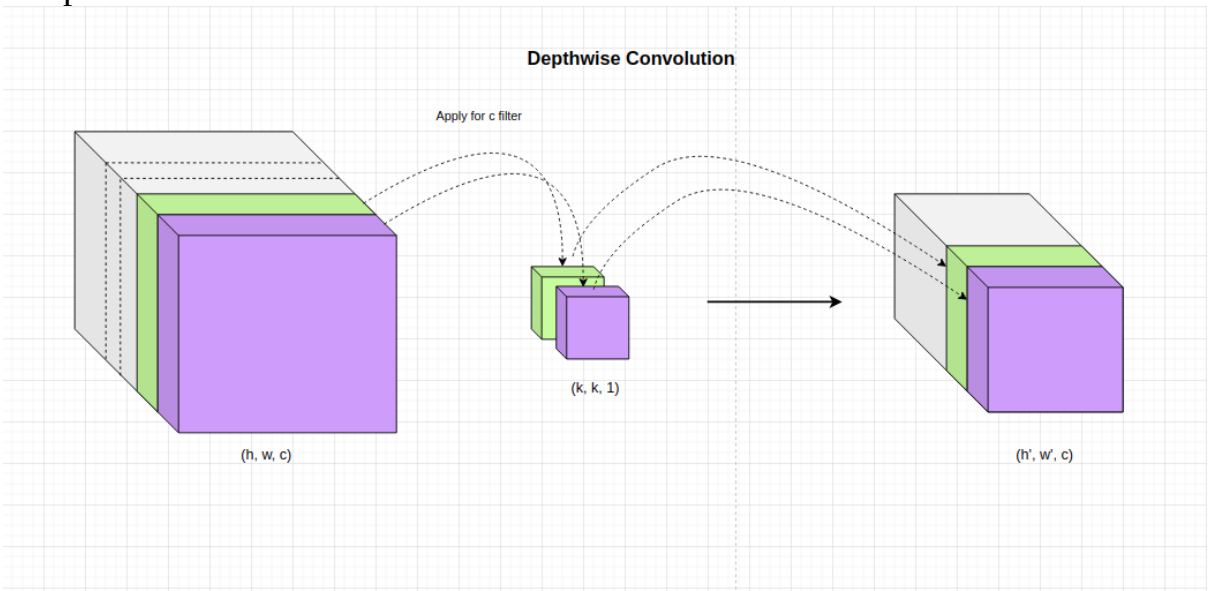
2.2.1 MobileNet

Mạng MobileNet là mạng tích chập nhưng được tinh chỉnh bằng một các kỹ thuật giúp cho mô hình này có thể giảm được các trọng số nhưng vẫn giữ được độ chính xác, cơ chế đó được gọi là Depthwise Separable

Depthwise Separable Chia CNN ra thành hai phần là deepwise Convolution và pointwise Convolution.



1. Deepwise Convolution

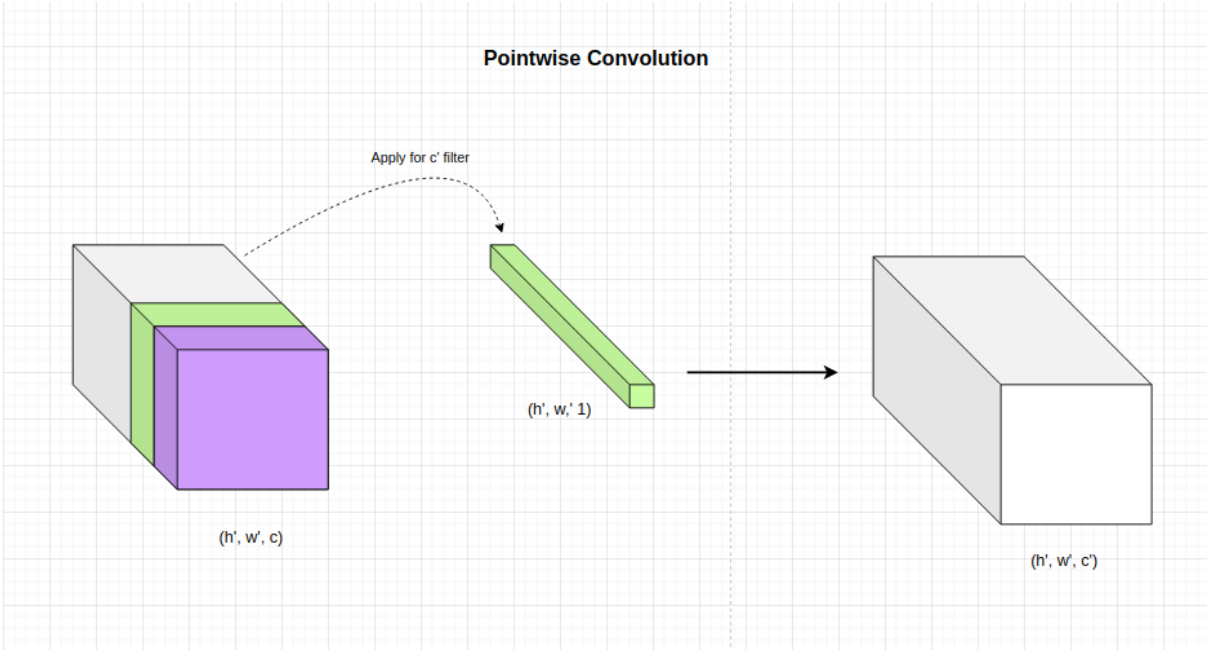


Mỗi một channel sẽ áp dụng một bộ lọc khác nhau và hoàn toàn không chia sẻ tham số. Điều này có ba tác dụng chính cho mô hình:

- Nhận diện đặc trưng: Quá trình học và nhận diện đặc trưng sẽ được tách biệt theo từng bộ lọc. Nếu đặc trưng trên các channels là khác xa nhau thì sử dụng các bộ lọc riêng cho channel sẽ chuyên biệt hơn trong việc phát hiện các đặc trưng. Chẳng hạn như đầu vào là ba kênh RGB thì mỗi kênh áp dụng một bộ lọc khác nhau chuyên biệt.
- Giảm thiểu khối lượng tính toán: Để tạo ra một điểm pixel trên output thì tích chập thông thường cần sử dụng k×k×c phép tính trong khi tích chập chiều sâu tách biệt chỉ cần k×k phép tính.
- Giảm thiểu số lượng tham số : Ở tích chập chiều sâu cần sử dụng c×k×k tham số. Số lượng này ít hơn gấp c′ lần so với tích chập chiều sâu thông thường.

Kết quả sau tích chập được concatenate lại theo độ sâu. Như vậy output thu được là một khối tensor3D có kích thước h′×w′×c

2. Pointwise Convolution: Có tác dụng thay đổi độ sâu của output bước trên từ c sang c'. Ta sẽ áp dụng c' bộ lục kích thước 1 x 1 x c. ⇒Kích thước width và height không thay đổi mà chỉ thay đổi độ sâu

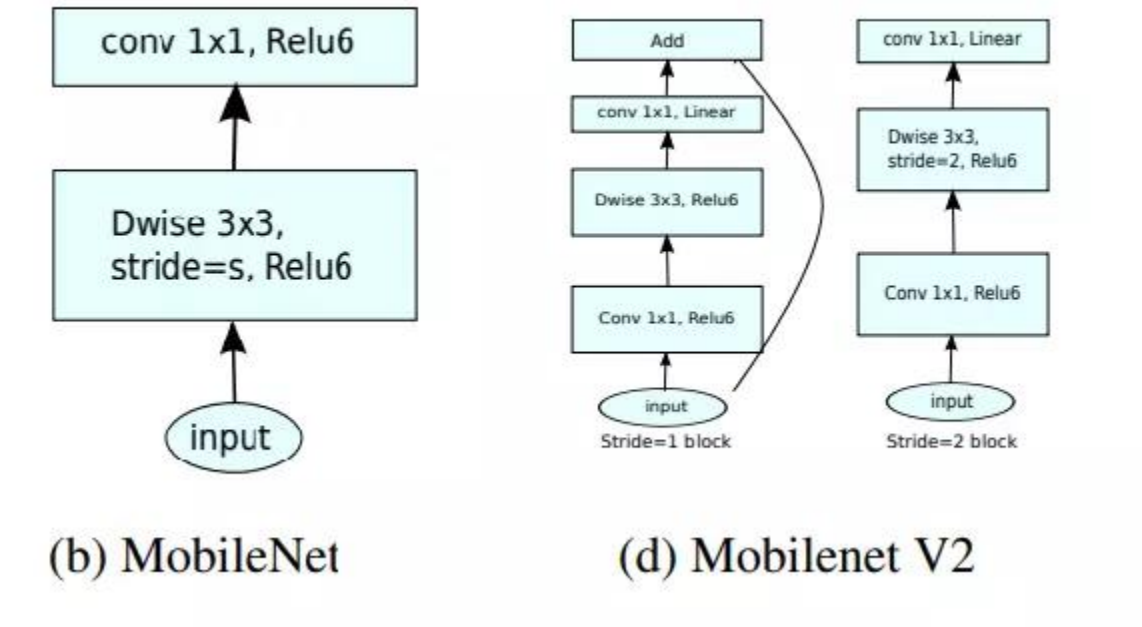


Kết quả sau cùng chúng ta thu được là một output có kích thước h′×w′×c′. Số lượng tham số cần áp dụng ở trường hợp này là c′×c

2.2.2 MobileNetV2:

MobileNet v2 tiếp tục sử dụng Depthwise Separable Convolutions, ngoài ra còn đề xuất thêm:

- Linear bottlenecks
- Inverted Residual Block (shortcut connections giữa các bottlenecks)



(b) MobileNet (d) Mobilenet V2

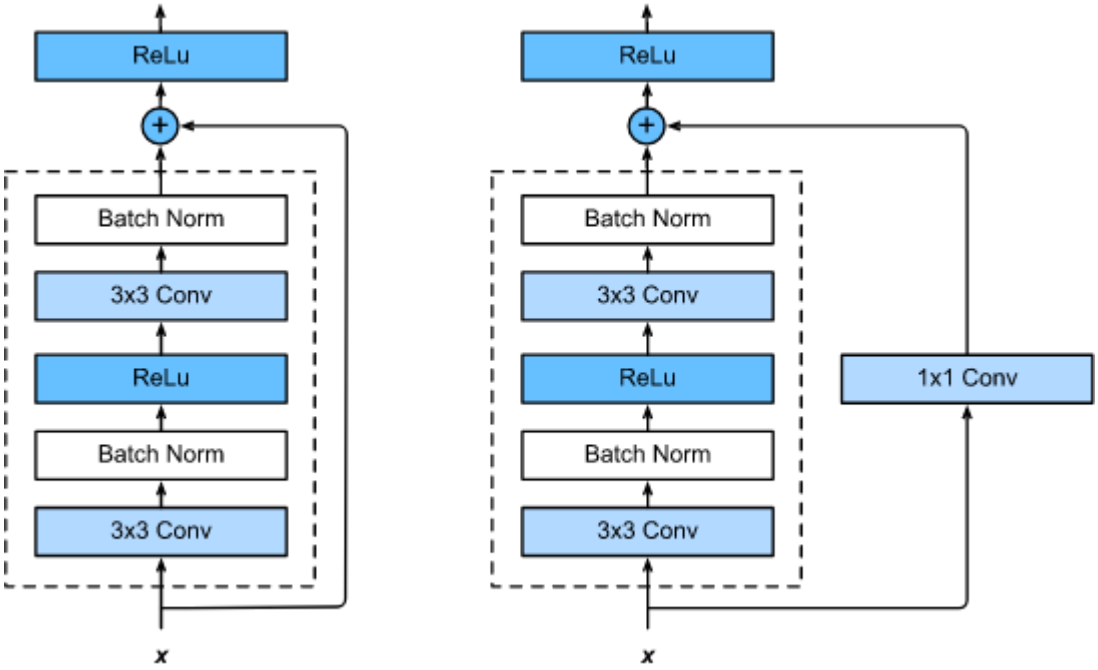
Hãy phân tích sự khác biệt trong bức ảnh trên:

- MobileNet gồm sử dụng 1 loại blocks gồm 2 phần, Deepwise và Pointwise.
- MobileNet v2 sử dụng 2 loại blocks, bao gồm: residual block với stride = 1 và block với stride = 2 phục vụ downsizing.

Có 3 phần đối với mỗi block:

- Layer đầu là 1×1 convolution với ReLU6.
- Layer thứ hai, như cũ, là depthwise convolution.
- Layer thứ 3 tiếp tục là 1×1 convolution nhưng không có activation function. Linear được sử dụng thay vì ReLu như bình thường.

MobileNetV2 cũng sử dụng những kết nối tắt như ở mạng ResNet:



Các khối ở layer trước được cộng trực tiếp vào layer liền sau. Nếu coi layer liền trước là x, sau khi đi qua các xử lý tích chập hai chiều ta thu được kết quả F(x) thì output cuối cùng là một residual block có giá trị x+F(x).

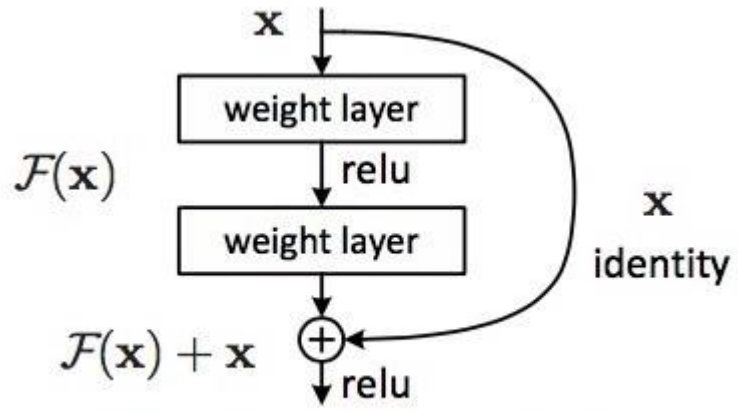


Figure 2. Residual learning: a building block.

Áp dụng trong bài toán nhận diện màu sắc hoa quả:

Thực hiện quá trình train mô hình

```
EPOCHS = 6
history = model.fit(train_generator,
                            steps_per_epoch=train_generator.n // BATCH_SIZE,
                            epochs=EPOCHS,
                            validation_data=val_generator,
                            validation_steps=val_generator.n // BATCH_SIZE,
                            verbose=1)
```

Lưu mô hình

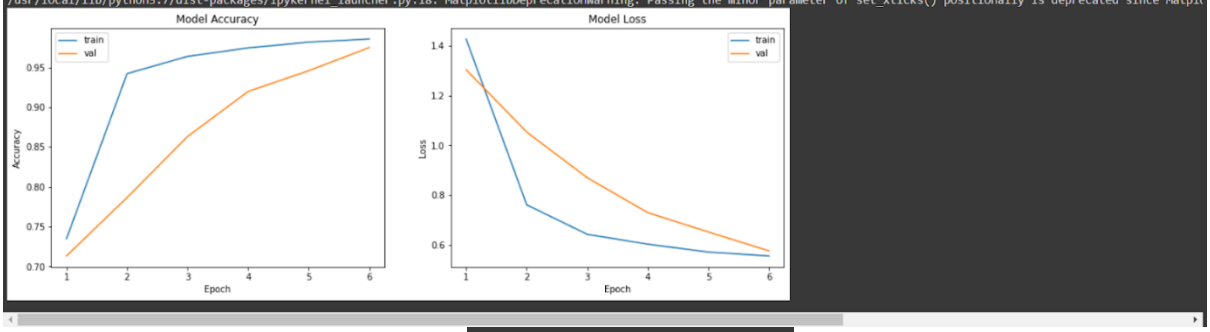
```
print(history)#in object lịch sử đào tạo
print(history.epoch) # in list epoch
print(history.history) #in lịch sử đào tạo
print(history.model) # in thông tin model
#Sử dụng Pickle để lưu các đối tượng
import pickle
filename = '/content/drive/MyDrive/fruit/train_history.pickle'
pickle.dump(history, open(filename, 'wb'))
```

```
<keras.callbacks.History object at 0x7fc13e181d10>
[0, 1, 2, 3, 4, 5]
{'loss': [1.4276834726333618, 0.7606154680252075, 0.6420339345932007, 0.6023861169815063, 0.570595920085907, 0.5550906658172607], 'accuracy': [0.7347051501274109, 0.9419581294059
<keras.engine.sequential.Sequential object at 0x7fc1b08a2450>
WARNING:absl:Function `_wrapped_model` contains input name(s) mobilenetv2_1.00_224_input with unsupported characters which will be renamed to mobilenetv2_1_00_224_input in the Sa
```

Đánh giá

```
import matplotlib.pyplot as plt
def plot_model_history(model_history):
    fig, axs = plt.subplots(1,2,figsize=(15,5))
    # vẽ đồ thị độ chính xác
    axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy']) #vẽ đồ thị accuracy
    axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_accuracy']) #vẽ đồ thị val_accuracy
    axs[0].set_title('Model Accuracy') #đặt tiêu đề đồ thị
    axs[0].set_ylabel('Accuracy') #đặt tiêu đề trục y
    axs[0].set_xlabel('Epoch') #đặt tiêu đề trục x
    axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.history['accuracy'])/10) #lấy vị trí và nhãn của trục x
    axs[0].legend(['train', 'val'], loc='best') #hiển thị ghi chú trong đồ thị
    # vẽ đồ thị độ tổn thất
    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss']) #vẽ đồ thị loss
    axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss']) #vẽ đồ thị val_loss
    axs[1].set_title('Model Loss') #đặt tiêu đề đồ thị
    axs[1].set_ylabel('Loss') #đặt tiêu đề trục y
    axs[1].set_xlabel('Epoch') #đặt tiêu đề trục x
    axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['loss'])/10) #lấy vị trí và nhãn của trục x
    axs[1].legend(['train', 'val'], loc='best') #hiển thị ghi chú trong đồ thị
    plt.show() #xuất đồ thị
plot_model_history(history)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: MatplotlibDeprecationWarning: Passing the minor parameter of set_xticks() positionally is deprecated since Matplo
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:18: MatplotlibDeprecationWarning: Passing the minor parameter of set_xticks() positionally is deprecated since Matplo
```



acc ≈ 97.4% và loss đạt min ≈ 0.5550906658172607

2.3 Xây dựng mô hình

Code:

Dữ liệu bao gồm(Train: ,Test:)
Gồm 6 nhãn là:

- rottenoranges
- rottenbanana
- rottenapples
- freshoranges
- freshbanana
- freshapples

Sử dụng MobileNetV2 trong module Tensorflow.keras và khởi tạo mô hình mobileNetV2

Untitled0.ipynb - Colaboratory (google.com)

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Flatten
from tensorflow.keras.models import Model
import pandas as pd


input_shape = (224, 224, 3)
mobilenet_model = MobileNetV2(include_top=False, weights='imagenet', input_shape=input_shape)
#output = mobilenet.layers[-1].output
#output = Flatten()(output)
#mobilenet_model = Model(mobilenet.input, output)

mobilenet_model.trainable = True
#fine_tune_at = 100
# Freeze all the layers before the `fine_tune_at` layer
#for layer in mobilenet_model.layers[:fine_tune_at]:
#    layer.trainable =  False

mobilenet_model.summary()
```

Tinh chỉnh mô hình với input mạng gồm 320 node và output là 6 node

```
model = Sequential()
model.add(mobilenet_model)
model.add(Dense(units=320, activation='relu', kernel_regularizer=l2(0.001)))
model.add(GlobalAveragePooling2D())
model.add(Dense(units=6, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.Adam(1e-5),
              metrics=['accuracy'])
model.summary()
```

Bản tóm tắt :

```
Found 10901 images belonging to 6 classes.
Found 2698 images belonging to 6 classes.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Funct  (None, 7, 7, 1280)       2257984
 ional)

 dense (Dense)               (None, 7, 7, 320)         409920

 global_average_pooling2d (G  (None, 320)              0
 lobalAveragePooling2D)

 dense_1 (Dense)             (None, 6)                 1926

=================================================================
Total params: 2,669,830
Trainable params: 2,635,718
Non-trainable params: 34,112
_________________________________________________________________
```

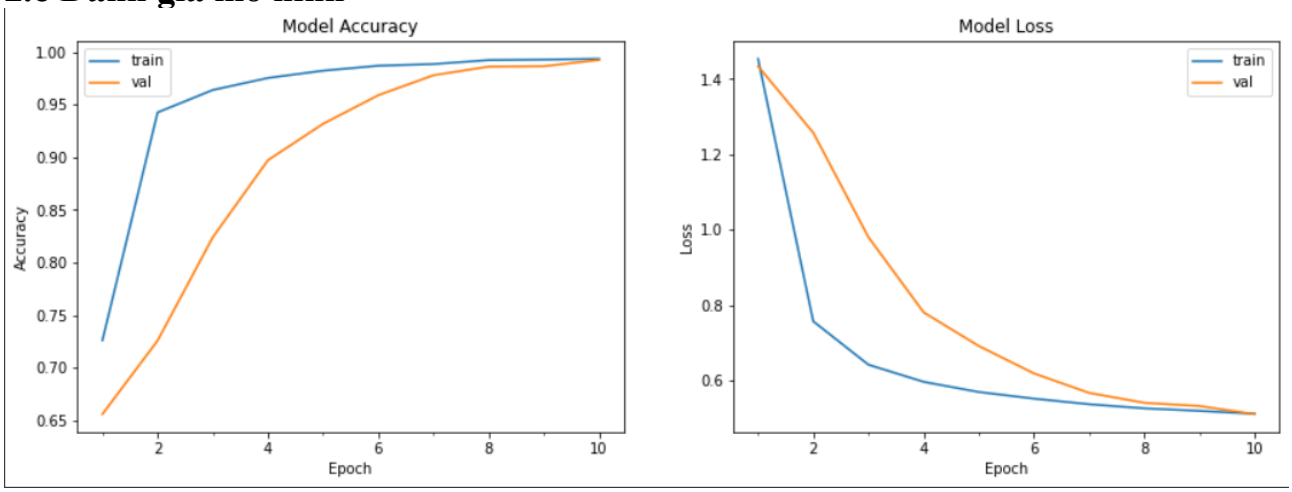
2.4 Quy trình đào tạo

```
[ ]  EPOCHS = 10
     history = model.fit(train_generator,
                                    steps_per_epoch=train_generator.n // BATCH_SIZE,
                                    epochs=EPOCHS,
                                    validation_data=val_generator,
                                    validation_steps=val_generator.n // BATCH_SIZE,
                                    verbose=1)
```

2.5 Hình dung quá trình đào tạo

```
Epoch 1/10
170/170 [==============================] - 184s 998ms/step - loss: 1.4532 - accuracy: 0.7260 - val_loss: 1.4336 - val_accuracy: 0.6559
Epoch 2/10
170/170 [==============================] - 168s 984ms/step - loss: 0.7570 - accuracy: 0.9427 - val_loss: 1.2573 - val_accuracy: 0.7258
Epoch 3/10
170/170 [==============================] - 167s 980ms/step - loss: 0.6418 - accuracy: 0.9640 - val_loss: 0.9802 - val_accuracy: 0.8240
Epoch 4/10
170/170 [==============================] - 168s 984ms/step - loss: 0.5965 - accuracy: 0.9755 - val_loss: 0.7808 - val_accuracy: 0.8973
Epoch 5/10
170/170 [==============================] - 168s 985ms/step - loss: 0.5694 - accuracy: 0.9824 - val_loss: 0.6915 - val_accuracy: 0.9319
Epoch 6/10
170/170 [==============================] - 168s 984ms/step - loss: 0.5518 - accuracy: 0.9871 - val_loss: 0.6188 - val_accuracy: 0.9591
Epoch 7/10
170/170 [==============================] - 168s 984ms/step - loss: 0.5370 - accuracy: 0.9887 - val_loss: 0.5671 - val_accuracy: 0.9781
Epoch 8/10
170/170 [==============================] - 167s 982ms/step - loss: 0.5261 - accuracy: 0.9924 - val_loss: 0.5408 - val_accuracy: 0.9862
Epoch 9/10
170/170 [==============================] - 167s 983ms/step - loss: 0.5194 - accuracy: 0.9929 - val_loss: 0.5323 - val_accuracy: 0.9866
Epoch 10/10
170/170 [==============================] - 168s 986ms/step - loss: 0.5117 - accuracy: 0.9936 - val_loss: 0.5107 - val_accuracy: 0.9926
```

2.6 Đánh giá mô hình



3. Lưu lịch sử đào tạo và mô hình

3.1 Lưu lịch sử đào tạo

```
import pickle
filename = '/content/drive/MyDrive/XLA/filemodel.pickle'
pickle.dump(history, open(filename, 'wb'))

```

```
WARNING:absl:Function `_wrapped_model` contains input name(s) mobilenetv2_1.00_224_input with unsupported characters which will be renamed to mobilenetv2_1_00_224_
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_
```

3.2 Lưu mô hình

-Trong phần thứ ba, 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.

```
[ ]  1 model.save('Fresh_Rotten_Fruits_MobileNetV2_Transfer_Learning.h5')
     2
```

4. Xây dựng mô hình phân loại độ xanh chín của hoa quả bằng thuật toán KMEAN

4.1. Giới thiệu về thuật toán KMEAN và các ứng dụng của nó

a, Giới thiệu:

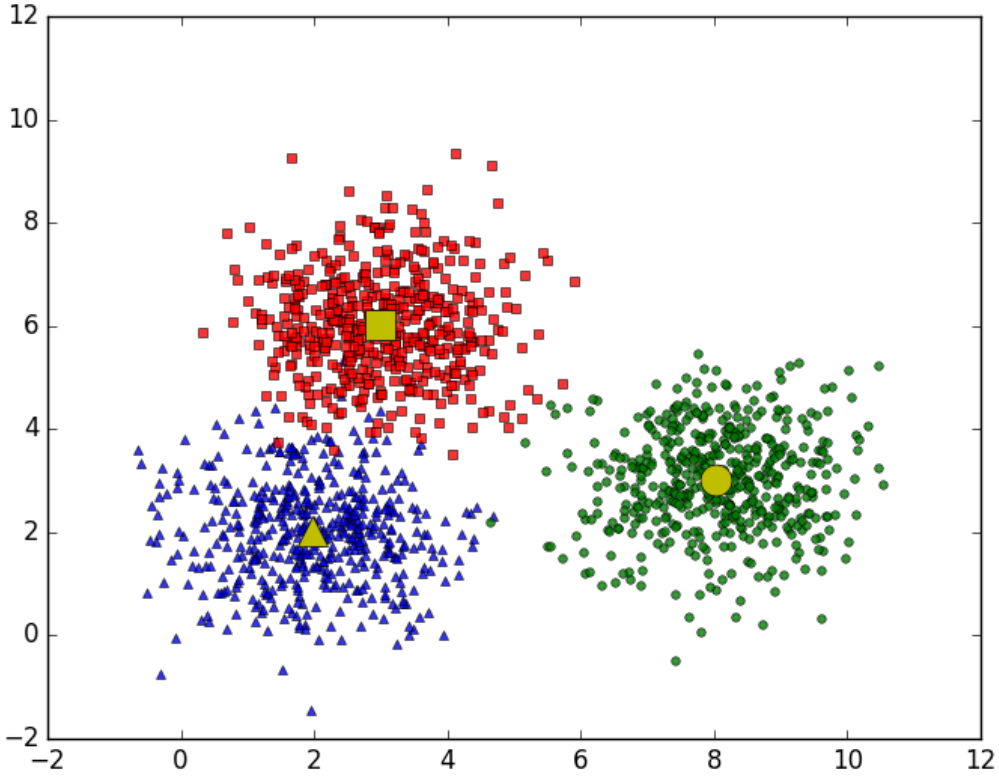
Thuật toán K-means clustering được ra đời để phân dữ liệu thành các cụm (cluster) khác nhau sao cho *dữ liệu trong cùng một cụm có tính chất giống nhau*.

Cụ thể, trong thuật toán k-Means mỗi cụm dữ liệu được đặc trưng bởi một *tâm* (*centroid*). *tâm* là điểm đại diện nhất cho một cụm và có giá trị bằng trung bình của toàn bộ các quan sát nằm trong cụm. Chúng ta sẽ dựa vào khoảng cách từ mỗi quan sát tới các *tâm* để xác định nhãn cho chúng trùng thuộc về *tâm* gần nhất. Ban đầu thuật toán sẽ khởi tạo ngẫu nhiên một số lượng xác định trước tâm cụm. Sau đó tiến hành xác định nhãn cho từng điểm dữ liệu và tiếp tục cập nhật lại tâm cụm. Thuật toán sẽ dừng cho tới khi toàn bộ các điểm dữ liệu được phân về đúng cụm hoặc số lượt cập nhật tâm chạm ngưỡng.

b, Ứng dụng thực tế:

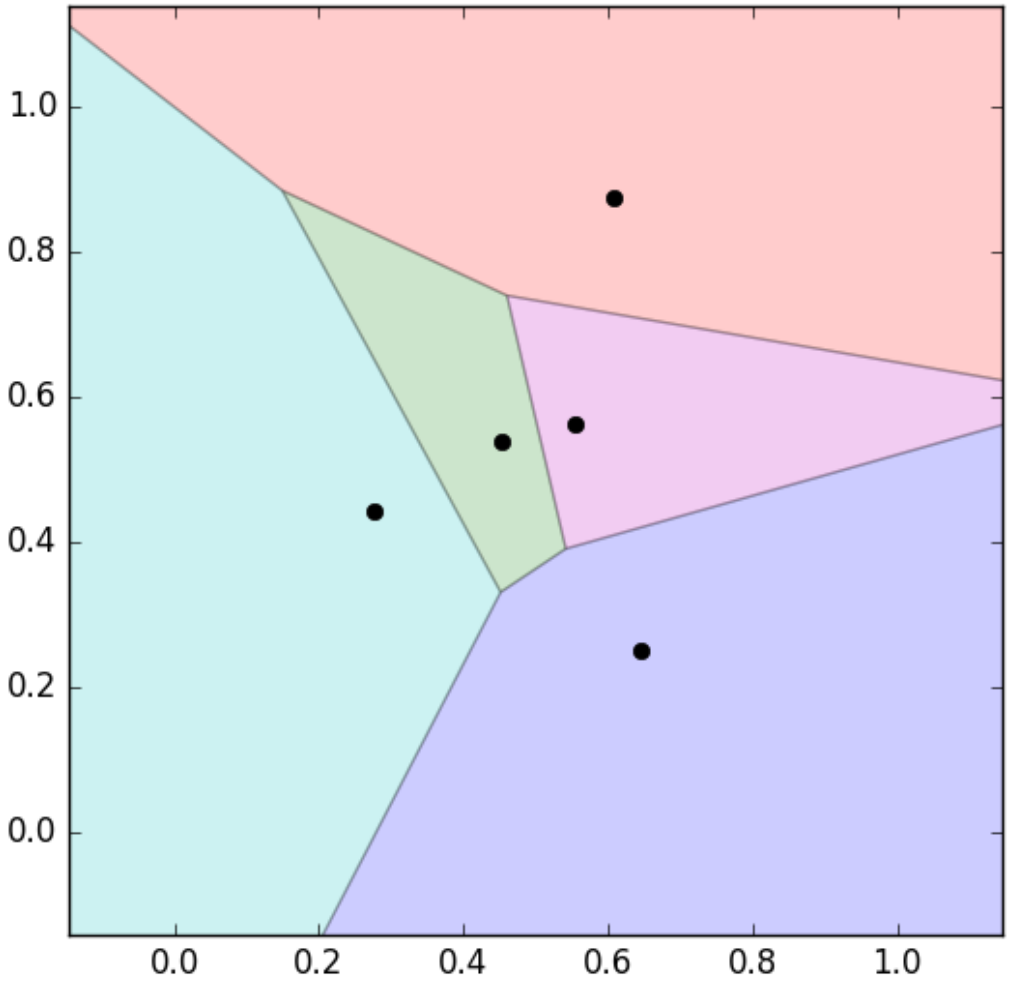
- Phân đoạn ảnh
- Nhận dạng
- Khai phá dữ liệu

Trong bài tập lớn này, chúng ta sẽ áp dụng K-means trong phân cụm màu:
Trong thuật toán K-means clustering, chúng ta không biết nhãn (label) của từng điểm dữ liệu. Mục đích là làm thể nào để phân dữ liệu thành các cụm (cluster) khác nhau sao cho *dữ liệu trong cùng một cụm có tính chất giống nhau*.
Ý tưởng đơn giản nhất về cluster (cụm) là tập hợp các điểm *ở gần nhau trong một không gian nào đó* (không gian này có thể có rất nhiều chiều trong trường hợp thông tin về một điểm dữ liệu là rất lớn). Hình bên dưới là một ví dụ về 3 cụm dữ liệu (từ giờ tôi sẽ viết gọn là *cluster*).



Giả sử mỗi cluster có một điểm đại diện (*center*) màu vàng. Và những điểm xung quanh mỗi center thuộc vào cùng nhóm với center đó. Một cách đơn giản nhất, xét một điểm bất kỳ, ta xét xem điểm đó gần với center nào nhất thì nó thuộc về cùng nhóm với center đó. Tới đây, chúng ta có một bài toán thú vị: *Trên một vùng biển hình vuông lớn có ba đảo hình vuông, tam giác, và tròn màu vàng như hình trên. Một điểm trên biển được gọi là thuộc lãnh hải của một đảo nếu nó nằm gần đảo này hơn so với hai đảo kia . Hãy xác định ranh giới lãnh hải của các đảo.*

Hình dưới đây là một hình minh họa cho việc phân chia lãnh hải nếu có 5 đảo khác nhau được biểu diễn bằng các hình tròn màu đen:



Chúng ta thấy rằng đường phân định giữa các lãnh hải là các đường thẳng (chính xác hơn thì chúng là các đường trung trực của các cặp điểm gần nhau). Vì vậy, lãnh hải của một đảo sẽ là một hình đa giác.

2. Các bước của thuật toán K-means:

Trong thuật toán k-Means mỗi cụm dữ liệu được đặc trưng bởi một *tâm* (*centroid*). *tâm* là điểm đại diện nhất cho một cụm và có giá trị bằng trung bình của toàn bộ các quan sát nằm trong cụm. Chúng ta sẽ dựa vào khoảng cách từ mỗi quan sát tới các *tâm* để xác định nhãn cho chúng trùng thuộc về *tâm* gần nhất. Ban đầu thuật toán sẽ khởi tạo ngẫu nhiên một số lượng xác định trước tâm cụm. Sau đó tiến hành xác định nhãn cho từng điểm dữ liệu và tiếp tục cập nhật lại tâm cụm. Thuật toán sẽ dừng cho tới khi toàn bộ các điểm dữ liệu được phân về đúng cụm hoặc số lượt cập nhật tâm chạm ngưỡng

Cụ thể các bước của thuật toán k-Means được tóm tắt như sau:

Đầu vào: Dữ liệu X và số lượng cluster cần tìm K.

Đầu ra: Các center M và label vector cho từng điểm dữ liệu Y.

1. Chọn K điểm bất kỳ làm các center ban đầu.
2. Phân mỗi điểm dữ liệu vào cluster có center gần nó nhất.

3. Nếu việc gán dữ liệu vào từng cluster ở bước 2 không thay đổi so với vòng lặp trước nó thì ta dừng thuật toán.
4. Cập nhật center cho từng cluster bằng cách lấy trung bình cộng của tất các các điểm dữ liệu đã được gán vào cluster đó sau bước 2.
5. Quay lại bước 2.

Từ các bước của thuật toán thì sau khi ảnh màu được cho qua thuật toán k-means sẽ được nhóm thành các nhóm màu với các xác suất khác nhau bằng thư viện sklearn và Kmeans bên trong module cluster

```
1   import cv2
2   import numpy as np
3   import matplotlib.pyplot as plt
4   from sklearn.cluster import KMeans
5   import imutils
6
7   clusters = 5 # try changing it
8
```

Tại dòng 1-5 sẽ khai báo các thư viện và dòng 7 khai báo biến clusters là số nhóm màu.

```
9   img = cv2.imread('5.png')
10  org_img = img.copy()
11  print('Org image shape --> ',img.shape)
12
13  img = imutils.resize(img,height=200)
14  print('After resizing shape --> ',img.shape)
15
```

Dòng 9: đọc hình ảnh đầu vào. Ở đấy lấy ảnh có tên là 5.png làm ví dụ.
Dòng 13: Thay đổi kích thước để tăng tốc độ xử lý với ảnh

```
16   flat_img = np.reshape(img,(-1,3))
17   print('After Flattening shape --> ',flat_img.shape)
18
```

Dòng 16: Làm phẳng, mục đích là giữ lại các cột của hình ảnh và để tạo ra một cột trong đó.

```
19   kmeans = KMeans(n_clusters=clusters,random_state=0)
20   kmeans.fit(flat_img)
```

Dòng 19,20: khởi tạo mô hình Kmeans trong module keras và fit vào ảnh (biến flat_img). Hình ảnh được làm phẳng đang hoạt động như một mảng chứa tất cả các màu pixel của hình ảnh. Các màu pixel này bây giờ sẽ được nhóm lại thành 5 nhóm. Các nhóm này sẽ có một số trung tâm mà chúng ta có thể coi là màu chính của cụm

```
22  dominant_colors = np.array(kmeans.cluster_centers_,dtype='uint')
23
24  percentages = (np.unique(kmeans.labels_,return_counts=True)[1])/flat_img.shape[0]
25  p_and_c = zip(percentages,dominant_colors)
26  p_and_c = sorted(p_and_c,reverse=True)
```

Dòng 22: Trích xuất các trung tâm của cụm. Bây giờ chúng ta biết rằng 5 màu này là màu chủ đạo của hình ảnh nhưng vẫn chưa biết mức độ thống trị của mỗi màu.
Dòng 24: Tính toán màu chủ đạo của ảnh.
np.unique(kmeans.labels_,return_counts=True) : Trả về mảng gồm 2 phần, phần đầu ra các dự đoán và phần sau sẽ là các mô tả giá trị trong pixel trong cụm và sau đó chia cho tổng số pixel trong hình ảnh (tổng các phần tử của phần thứ 2 trong mảng).
Dòng 25: Nén các số thành tỉ lệ phần trăm và mày sắc
Dòng 26: Sắp xếp lại theo xác suất màu nhiều nhất giảm dần

4.2. Xây dựng thuật toán tìm màu gần nhất

```
def closest_colour(requested_colour):
    min_colours = {}
    for key, name in webcolors.CSS3_HEX_TO_NAMES.items():
        r_c, g_c, b_c = webcolors.hex_to_rgb(key)
        rd = (r_c - requested_colour[0]) ** 2
        gd = (g_c - requested_colour[1]) ** 2
        bd = (b_c - requested_colour[2]) ** 2
        min_colours[(rd + gd + bd)] = name
    return min_colours[min(min_colours.keys())]
```

```
def get_colour_name(requested_colour):
    try:
        closest_name = actual_name = webcolors.rgb_to_name(requested_colour)
    except ValueError:
        closest_name = closest_colour(requested_colour)
        actual_name = None
    return actual_name, closest_name
```

```
def name_main_color(list):
    """
    find color near (Euclidean distance)
    input: code RGB
    output: RGB near
    """

    RED= ["lighsalmon","salmon","darksalmon","LightCoral","IndianRed","Crimson","Red","FireBrick","DarkRed"]
    ORANGE= ["Orange","DarkOrange","Coral","Tomato","OrangeRed"]
    YELLOW= ["Gold","Yellow","LightYellow","LemonChiffon","LightGoldenRodYellow","PapayaWhip","Moccasin","PeachPuff","PaleGoldenRod","Khaki","DarkKhaki"]
    GREEN= ["GreenYellow","Chartreuse","LawnGreen", "Lime","LimeGreen","PaleGreen","LightGreen","MediumSpringGreen","SpringGreen","MediumSeaGreen","SeaGreen","ForestGreen","Green",


    res="UNKNOWN"
    for i in list:
        color=get_colour_name(i)[1]
        if color.capitalize() in RED or color.capitalize() in ORANGE or color.capitalize() in YELLOW :
            res="CHÍN"
            break
        if color in GREEN:
            res="XANH"
            break


    return res
```

4.3 Xây dựng thuật toán Kmeans

```
def color_of_image(filepath):
  clusters = 3 # try changing it
  img = cv2.imread(filepath)
  org_img = img
  img = imutils.resize(img,height=200)
  flat_img = np.reshape(img,(-1,3))
  kmeans = KMeans(n_clusters=clusters,random_state=0)
  kmeans.fit(flat_img)
  dominant_colors = np.array(kmeans.cluster_centers_,dtype='uint')
  percentages = (np.unique(kmeans.labels_,return_counts=True)[1])/flat_img.shape[0]

  p_and_c = zip(percentages,dominant_colors)
  p_and_c = sorted(p_and_c,reverse=True)

  image=[]
  for i in range(len(p_and_c)):
    image.append(p_and_c[i][1])
    image[i]=(image[i].tolist())
    image[i].reverse()

  rows = 1000
  cols = int((org_img.shape[0]/org_img.shape[1])*rows)
  img = cv2.resize(org_img,dsize=(rows,cols),interpolation=cv2.INTER_LINEAR)

  copy = img
  cv2.rectangle(copy,(rows//2-250,cols//2-90),(rows//2+100,cols//2+110),(255,255,255),-1)

  final = cv2.addWeighted(img,0.1,copy,0.9,0)
  cv2.putText(final,'Most Dominant Colors',(rows//2-230,cols//2-40),cv2.FONT_HERSHEY_DUPLEX,0.8,(0,0,0),1,cv2.LINE_AA)


  start = rows//2-220
  for i in range(3):
      end = start+70
      final[cols//2:cols//2+70,start:end] = p_and_c[i][1]
      cv2.putText(final,str(i+1),(start+25,cols//2+45),cv2.FONT_HERSHEY_DUPLEX,1,(255,255,255),1,cv2.LINE_AA)
      start = end+20

  target_img = os.path.join(os.getcwd() , 'static\images')
  unique_filename = str(uuid.uuid4())
  filename = unique_filename+".jpg"
  img_path_kmean = os.path.join(target_img , filename)

#   cv2.imshow("oke",final)
  cv2.waitKey(0)
  cv2.destroyAllWindows()
  cv2.imwrite(img_path_kmean,final)
```

5. Đẩy mô hình lên web và thử nghiệm:

5.1. Giới thiệu sơ qua về Flask:

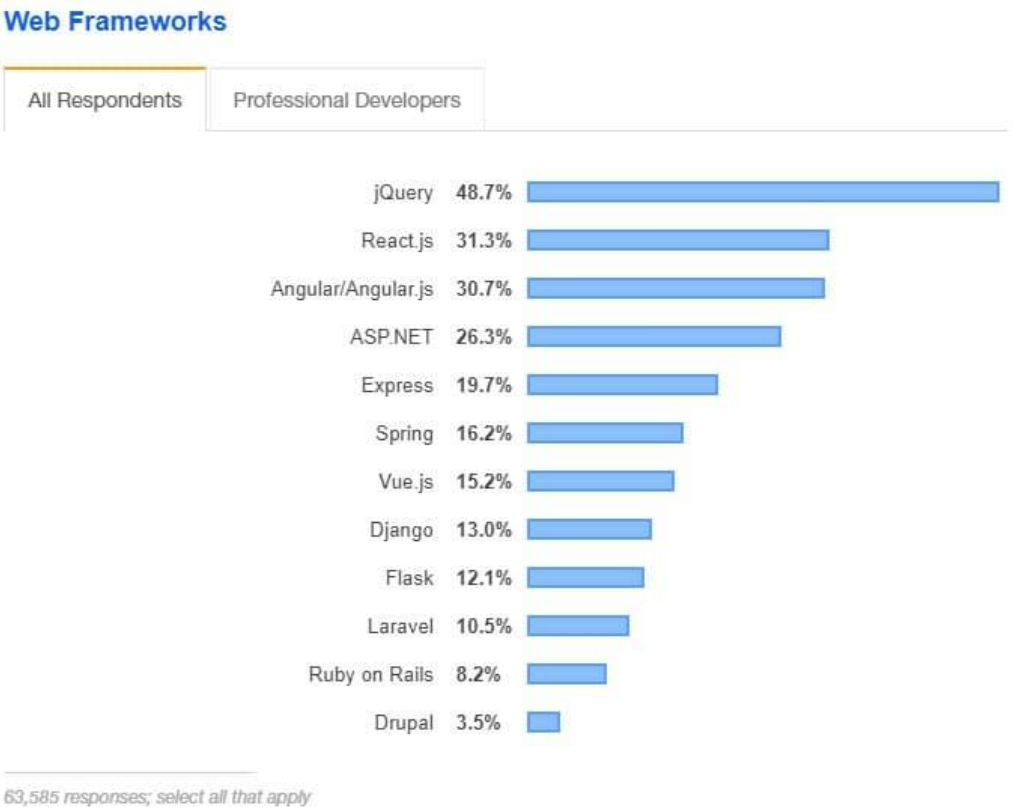
5.1.1. Python Flask là gì?

Flask là một Web Framework rất nhẹ của Python, dễ dàng giúp người mới bắt đầu học Python có thể tạo ra website nhỏ. Flask cũng dễ mở rộng để xây dựng các ứng dụng web phức tạp.

Flask là gì?

Flask là gì? Tìm hiểu về Flask Framework

Flask có nền tảng là Werkzeug và Jinja2 và nó đã trở thành một trong những Web Framework phổ biến nhất của Python

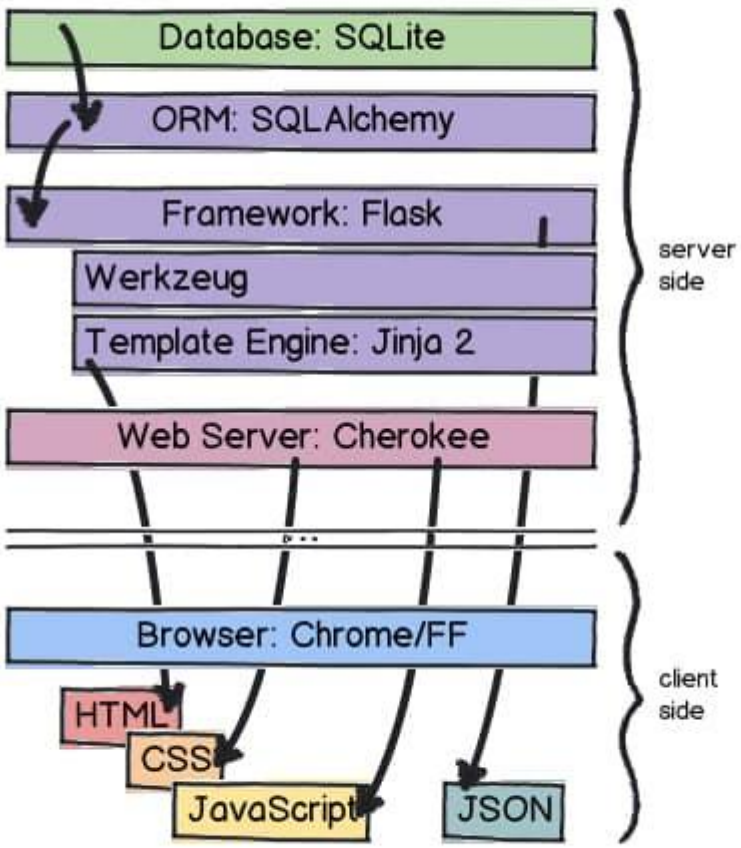


Web Framework phổ biên nhất thế giới theo

Thậm chí Flask còn phổ biến hơn cả Laravel.

Là một lập trình viên đã lập trình nhiều website bằng python, mình chắc chắn rằng bạn có thể sử dụng Flask để gia tăng lợi thế của mình.

Flask Framework là một bộ lưu trữ giúp các lập trình viên tạo ra các trang web dễ dàng hơn, có thể mở rộng, hiệu quả và có thể bảo trì bằng cách cung cấp code hoặc tiện ích mở rộng có thể sử dụng lại cho các nhiệm vụ phổ biến.



Cách Flask Framework hoạt động

5.1.2.Tính năng của Flask Framework

- Phát triển máy chủ
- Phát triển trình gỡ lỗi
- Hỗ trợ sẵn sàng để kiểm thử đơn vị
- Jinja2 templates
- RESTful request dispatch
- Hỗ trợ bảo mật cookie
- Full WSGI compliant
- Tài liệu mở rộng
- Dựa trên Unicode
- Khả năng tương thích công cụ dựa trên ứng dụng Google
- Nhiều tiện ích mở rộng cho các tính năng mong muốn
- Tính modular và thiết kế gọn nhẹ
- ORM-agnostic
- Độ linh hoạt cao
- Cung cấp xử lý HTTP request
- API có độc đáo và mạch lạc
- Dễ dàng triển khai

Tại sao sử dụng Flask?

-Flask cung cấp cho các lập trình viên khả năng tùy biến khi phát triển ứng dụng web, nó cung cấp cho bạn các công cụ, thư viện và cơ chế cho phép bạn xây dựng một ứng dụng web nhưng nó sẽ không thực thi bất kỳ sự phụ thuộc nào hoặc cho bạn biết dự án sẽ như thế nào.

-Ứng dụng web có thể là blog, trang web thương mại hoặc một số trang web khác, nó vẫn cho phép các lập trình viên cơ hội sử dụng một số tiện ích mở rộng để thêm nhiều chức năng hơn cho ứng dụng web.

-Cụ thể đối với project này, quá trình huấn luyện mô hình mới chỉ tạo ra các sản phẩm chạy được trên jupyter notebook. Có một AI engineer khá nổi tiếng nói rằng: model trên jupyter notebook là model chết. Đúng vậy, vì nếu không đưa sản phẩm lên production thì mọi việc chúng ta làm đều unusable.

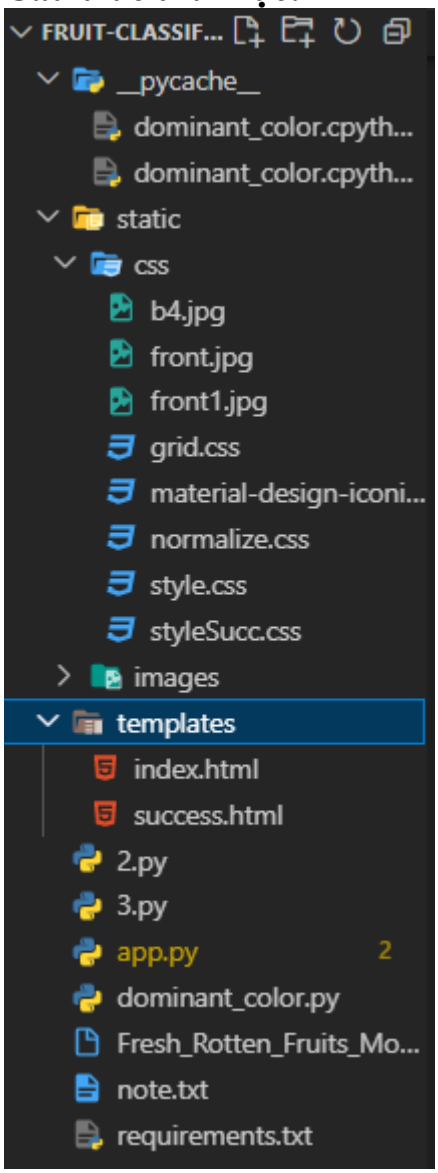
-Với các doanh nghiệp lớn, để đưa được model vào ứng dụng sẽ cần quá trình POC, DEV, stress test, QC, deploy. Các quá trình như stress test, QC, deploy sẽ không

được thực hiện bởi data scientist mà được thực hiện bởi các data engineer, QA. Chính vì thế khả năng triển khai một ứng dụng lên production luôn là một điểm hạn chế của data scientist.

-Vì vậy để triển khai modal phân loại của project trở thành một sản phẩm có thể sử dụng dễ dàng trên giao diện website chúng em đã lựa chọn framework Flask

5.2 Hướng dẫn chạy project

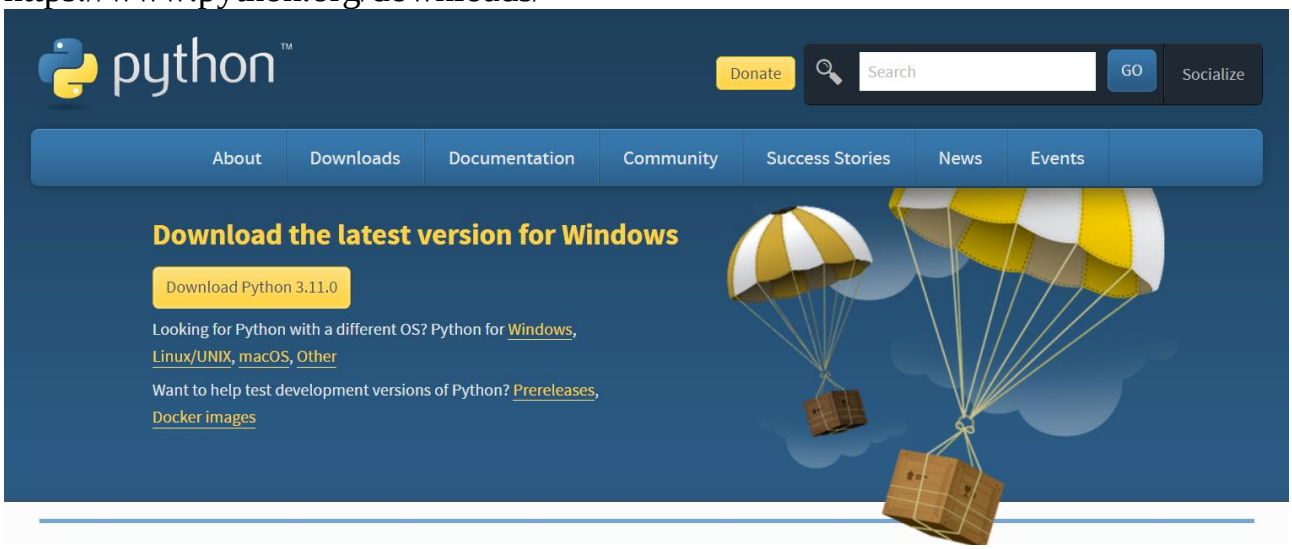
Cấu trúc thư mục:



Cách chạy :

Cài đặt môi trường Python

https://www.python.org/downloads/



Cài đặt các thư viện được sử dụng cho project

```
lenovo@DESKTOP-591I848 MINGW64 ~/Desktop/fruit-classification
$ pip install requirements.txt
```

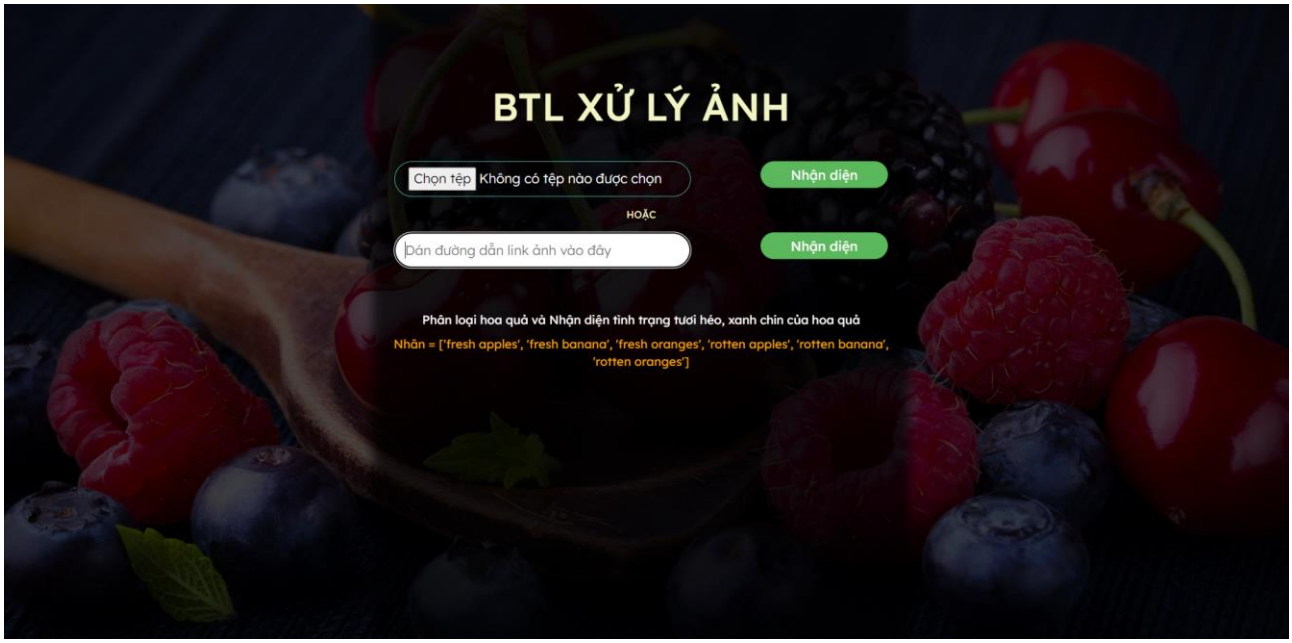
Bắt đầu khởi chạy project

```
lenovo@DESKTOP-591I848 MINGW64 ~/Desktop/fruit-classification
$ py app.py
```

Terminal thông báo đã khởi chạy thành công và có thể truy cập website với URL: http://127.0.0.1:5000

```
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
2022-11-23 11:42:09.148909: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-23 11:42:09.161122: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2022-11-23 11:42:16.687814: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found
2022-11-23 11:42:16.700639: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-11-23 11:42:16.717712: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-591I848
2022-11-23 11:42:16.732492: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-591I848
2022-11-23 11:42:16.741498: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructio
ns in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Debugger is active!
 * Debugger PIN: 814-235-736
```

5.3 . Test thử nghiệm



Test thử 1 ảnh thứ nhất :

Test thử ảnh thứ 2 :

Test thử ảnh thứ 3: