ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO ĐỒ ÁN

THỊ GIÁC MÁY TÍNH NÂNG CAO

CS331.M21.KHCL

***ĐỀ TÀI*: PHÂN LOẠI THỜI TIẾT**

GIẢNG VIÊN HƯỚNG DẪN: TS. MAI TIẾN DŨNG

SINH VIÊN THỰC HIỆN: LÊ NGUYỄN TIẾN ĐẠT– MSSV: 20521167

TP. HỒ CHÍ MINH, 2/2022

MỤC LỤC

I.Giới thiệu chung về đề tài

1.Lý do chọn đề tài

Việc nhận diện được các hiện tượng thời tiết thường có ảnh hưởng rất lớn tới nhiều khía cạnh khác nhau trong đời sống của chúng ta, ví dụ như dự báo thời tiết, đánh giá tình trạng của những con đường, giao thông, nông nghiệp, quản lý rừng bền vững và việc nhận diện các môi trường tự nhiên. Tuy nhiên, hiện nay lại có rất ít đề tài hay bài báo hướng đến việc phân loại những hiện tượng tự nhiên, chủ yếu quan san bằng mắt thường của con người.

Vì vậy, đề tài mà em hướng tới trong đồ án môn học này là vô cùng cần thiết tới cuộc sống thường ngày. Nhóm chúng em có đưa ra hai phương pháp để giải quyết vấn đề trên, cụ thể đó là một model CNN được điều chỉnh một số chức năng cho phù hợp với bài toán, hai là sử dụng model VGG16. Chúng em chọn 2 phương pháp này chủ yếu vì lý do như sau:

- CNN là mô hình để phân lớp rất hiệu quả trên 1 tập dữ liệu lớn,nên nó có khả năng cho 1 kết quả vô cùng chính xác.
- VGG16 là một model từng đạt giải thưởng cao trong cuộc thi phân loại ảnh vào năm 2015, nó mang lại độ chính xác cao và thời gian chạy vô cùng tốt nên được áp dụng nhiều vào các bài toán nhận diện và phân loại ảnh.

2.Input và output

+ Input: tập data gồm 5 loại thời tiết gồm rainy (mưa), cloudy (nhiều mây), foggy (có sương mù), shine (chói sáng), sunrise (mặt trời mọc) từ tập data Multi-class weather dataset (MWD). Các bức ảnh chỉ gồm một hiện tượng thời tiết duy nhất.
+ Output: Là nhãn của các hiện tượng thời tiết như sau: Rainy, Cloudy, Foggy, Shine, Sunrise.

3.Bộ Data

Bộ data Multi-class Weather Dataset (MWD) là bộ data lớn được sử dụng trong bài báo nghiên cứu mang tên "Multi-class weather recognition from still image using heterogeneous ensemble method". Bộ dataset cung cấp một tập các bức ảnh về các hiện tượng thời tiết được trích xuất bởi nhiều phương thức khác nhau để có thể dự đoán được nhiều điều kiện thời tiết khác nhau.

II.Giới thiệu về kỹ thuật Data Augmentation

Với mục đích tăng cường dữ liệu cho tập train, nhóm em đã sử dụng kỹ thuật Data augmentation, giúp tăng cường dữ liệu thông qua một số phép biến đổi, giúp ngăn chặn overfitting và làm cho model được khái quát hóa một cách tốt hơn.

Trong Keras ta có thể sử dụng Data Augmentation bằng dòng lệnh `keras.preprocessing.image.ImageDataGenerator.` Câu lệnh này cho phép bạn:

1. Cấu hình các dạng chuyển đổi ngẫu nhiên và chuẩn hóa sẽ được thực hiện trên dữ liệu hình ảnh trong quá trình train.
2. Khởi tạo các ảnh tăng cường (và nhãn của chúng) thông qua `.flow(data, labels)` hoặc `.flow_from_directory(directory)`. Các bộ khởi tạo này sau

đó được sử dụng với mô hình Keras dùng các bộ tạo dữ liệu đầu vào như `fit_generator`

```
train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=40,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)
```

Ta sẽ load dataset với image generator từ Keras. Ở đây, chúng ta sẽ cấu hình như sau: thay đổi kích thước, xoay, cắt ảnh theo 1 phạm vi, thu nhỏ phóng to và lật ngang.

```
BATCH_SIZE = 32
IMAGE_SIZE = 150

train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=40,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)
validation_datagen = ImageDataGenerator(rescale=1./255)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        '/content/369-1/train',
        target_size=(IMAGE_SIZE, IMAGE_SIZE),
        batch_size=BATCH_SIZE,
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
        '/content/369-1/valid',
        target_size=(IMAGE_SIZE, IMAGE_SIZE),
        batch_size=BATCH_SIZE,
        class_mode='categorical')

test_generator = validation_datagen.flow_from_directory(
        '/content/369-1/test',
        target_size=(IMAGE_SIZE, IMAGE_SIZE),
        batch_size=BATCH_SIZE,
        class_mode='categorical')
```

III.Giới thiệu về Model CNN

1.Khái niệm

CNN hay convolutional neural network (CNN) là 1 phương thức của mạng học sâu (Deep Neural Network). CNN là 1 thuật toán Machine Learning mà đầu vào là một bức ảnh cùng với một số dữ liệu khác như weights hay biases hay nhiều vật thể có trong một bức ảnh và đầu ra của nó có thể phân biệt vật thể này với những vật thể còn lại.

CNN hoạt động bằng cách trích xuất features từ những bức ảnh. Bất cứ CNN nào cũng bao gồm các bước sau đây:

1. Lớp Input là một bức ảnh xám hoặc ảnh màu.
2. Lớp Output là 2 hay nhiều nhãn chưa class
3. Lớp Hidden bao gồm nhiều lớp convolution, lớp ReLU, lớp pooling và một fully-connected Neural Network.

Ở đồ án này, chúng ta sẽ thiết kế một mạng Neural Network dùng để nhận diện các hiện tượng thời tiết sử dụng một dataset để train bao gồm 3348 bức ảnh với 5 loại hiện tượng thời tiết (rainy, cloudy, sunrise, shine, foggy).

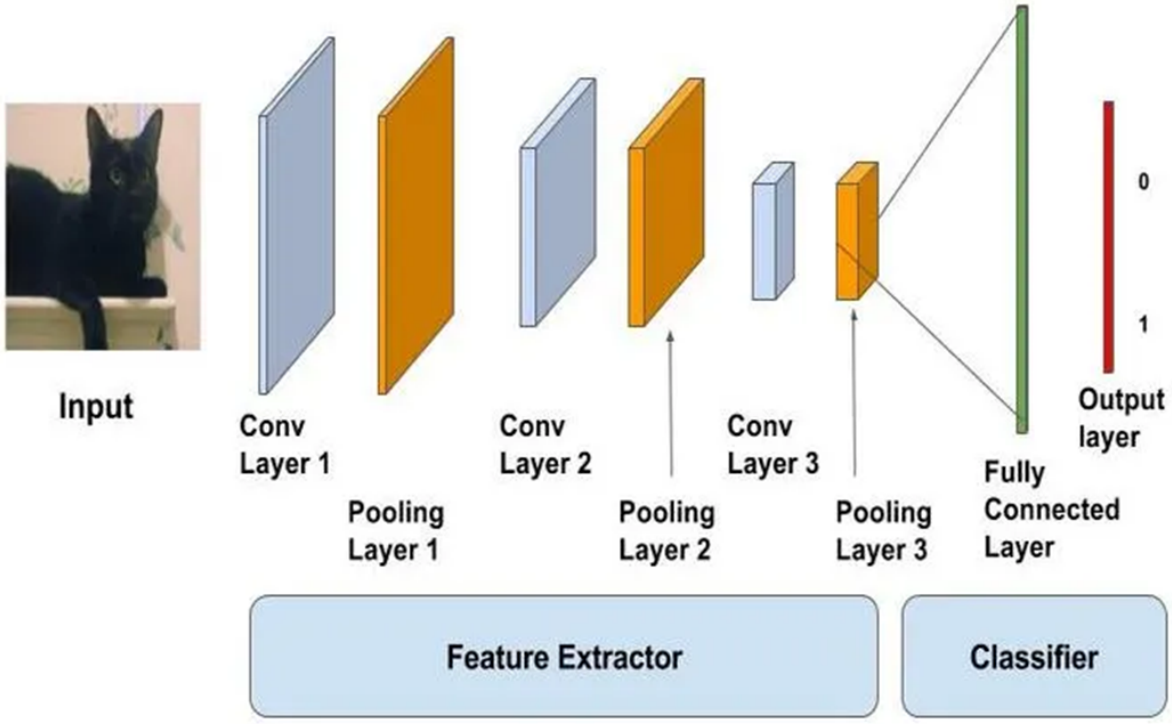
2. Áp dụng thuật toán CNN vào bài toán

Model CNN bao gồm có 2 bước chính: Feature extraction và classification.

Feature extraction là giai đoạn mà các filters và lớp sẽ được dùng để trích xuất thông tin từ những bức ảnh và một khi hoàn thành hết các bước trên sẽ tự chuyển tới giai đoạn tiếp theo là classification nơi mà những bức ảnh sẽ được phân loại dựa vào các nhãn input của bài toán.

Một mạng CNN thường sẽ như sau:

1. Input layer
2. Convolution layer + Activation function
3. Pooling layer
4. Fully – Connected Layer



Ở đồ án này chúng ta sẽ áp dụng một model CNN đơn giản với 2 CNN blocks gọi là Model 2.

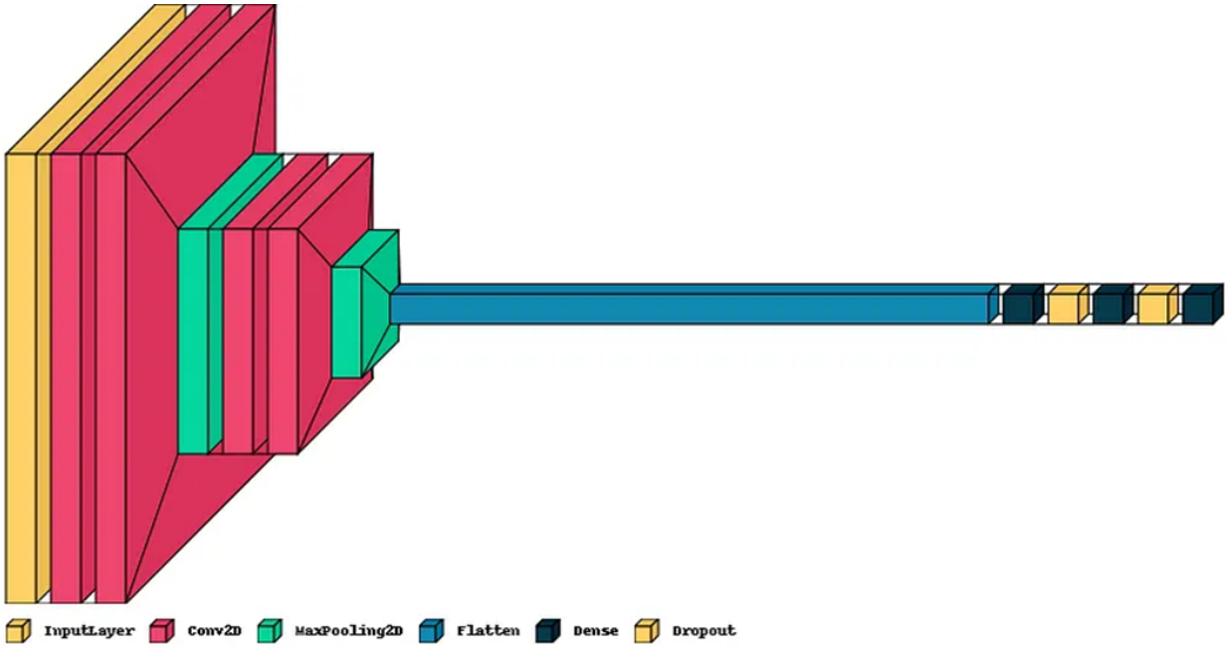
Kiến trúc của Model 2

Chúng ta sẽ sử dụng 2 khối convolutional như sau:

1. Khối 1: hai lớp 32 filters, theo sau là 1 lớp Max-pooling.
2. Khối 2: hai lớp 64 filters, theo sau là 1 lớp Max-pooling.
3. Fully – connected: 2 lớp Dense và 2 lớp dropout, cuối cùng là 1 lớp classify sử dụng hàm kích hoạt “softmax”.

```
keras.backend.clear_session() #clear model numbers

input_layer = keras.layers.Input(shape=(IMAGE_SIZE,IMAGE_SIZE,3))

#CNN - Convolutional Layers
conv_layer = keras.layers.Conv2D(32, (3,3), activation="relu", padding="same")(input_layer)
conv_layer = keras.layers.Conv2D(32, (3,3), activation="relu", padding="same")(conv_layer)
pooling_layer = keras.layers.MaxPool2D(pool_size=(2,2))(conv_layer)

conv_layer = keras.layers.Conv2D(64, (3,3), activation="relu", padding="same")(pooling_layer)
conv_layer = keras.layers.Conv2D(64, (3,3), activation="relu", padding="same")(conv_layer)
pooling_layer = keras.layers.MaxPool2D(pool_size=(2,2))(conv_layer)

#FC - Fully connected
flatten = keras.layers.Flatten()(pooling_layer)
dense = keras.layers.Dense(200, activation="relu")(flatten)
dropout = keras.layers.Dropout(0.5)(dense)
dense = keras.layers.Dense(100, activation="relu")(dropout)
dropout = keras.layers.Dropout(0.2)(dense)

classifier = keras.layers.Dense(5, activation="softmax")(dropout)

model2 = keras.Model(inputs=input_layer, outputs=classifier)
opt = keras.optimizers.Adam(learning_rate=0.001)
model2.compile(optimizer=opt, loss="categorical_crossentropy",metrics=['accuracy'])
model2.summary()
```

Ta trực quan hóa kiến trúc của Model 2 như sau:



IV. Giới thiệu về Transfer Learning

1.Khái niệm Transfer Learning

Trước đây ta có thể mất hằng giờ để train một neural network trên các tập dataset lớn. Tuy nhiên, hiện nay thời gian này có thể được rút ngắn nhờ các weights từ các pretrained model được train từ trước. Nói cách khác, chúng ta sẽ áp dụng Transfer learning vào bài toán này.

Transfer Learning là một kỹ thuật hoạt động vô cùng hiệu quả trong các lĩnh vực như phân loại ảnh và xử lý ngôn ngữ tự nhiên.

Convolutional Network rất tốt cho việc xử lý các vấn đề về hình ảnh. Tuy nhiên, nó lại cực kì tốn kém tài nguyên máy nếu ta sử dụng một kiến trúc mạng lớn cùng với việc không có một GPU đủ mạnh. Từ đây, ta sẽ có giải pháp:

1. Lấy các lớp từ một pretrained model.

2. Đóng băng các lớp đó để tránh việc bị mất bất cứ thông tin nào mà lớp đó chứa trong quá trình train.
3. Thêm vào một số lớp mới, có thể huấn luyện chúng trên các lớp bị đóng băng. Chúng sẽ học để chuyển các feature cũ thành dự đoán trên bộ dataset mới.
4. Huấn luyện các lớp mới trên tập dataset của chúng ta.

2. VGG16

AlexNet được ra đời vào năm 2012 và nó được cải tiến từ các convolutional neural network truyền thống, vì vậy chúng ta có thể hiểu VGG là sản phẩm kế thừa của AlexNet nhưng nó được tạo nên bởi một group khác với tên gọi là Visual Geometry Group ở Oxford, do đó nó có tên VGG. Nó mang và sử dụng một số ý tưởng từ những sản phẩm tiền nhiệm của nó và cải thiện chúng đồng thời nó sử dụng deep convolutional neural layers để cải thiện độ chính xác.

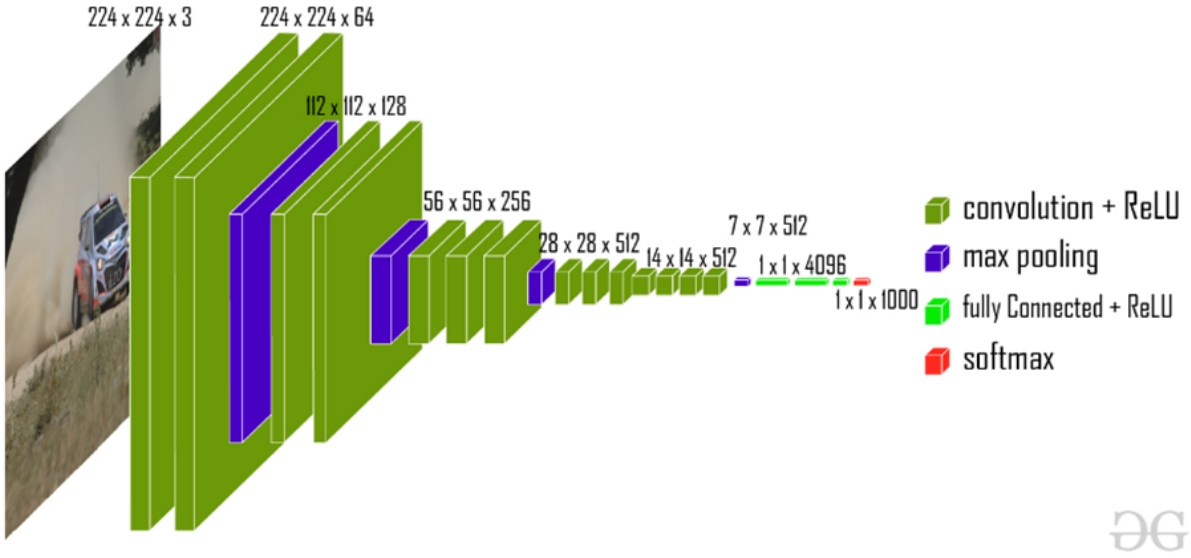
Cấu trúc VGG

Đầu vào của mạng là hình ảnh có kích thước (224,224,3). Hai lớp đầu tiên có 64 channels với bộ lọc có kích thước 3*3 và cùng một padding. Sau đó một lớp max pool với stride (2,2), hai convolutinal layers có 128 bộ lọc với kích thước (3,3). Tiếp theo là lớp max-pooling với stride (2,2) giống như lớp trước đó. Sau đó, có 2 convolutional layers với kích thước bộ lọc (3,3) và 256 bộ lọc. Sau đó có 2 bộ với 3 convolutional layers và max pool layer. Mỗi bộ lọc trong 512 bộ lọc có kích thước (3,3) với cùng một padding. Hình ảnh này sau đó được chuyển đến stack của convolutional layer. Trong các convolutional layer và max-pooling layer, các bộ lọc được sử dụng với kích thước 3*3 thay vì 11*11 như trong AlexNet và 7*7 trong ZF-Net. Trong một số lớp, nó cũng sử dụng pixel 1*1, được sử dụng để thao tác với số channels đầu vào. Có một padding với 1 pixel (same padding) được thực hiện sau mỗi convolutional layer để ngăn những đặc trưng không gian của hình ảnh.

VGG-16

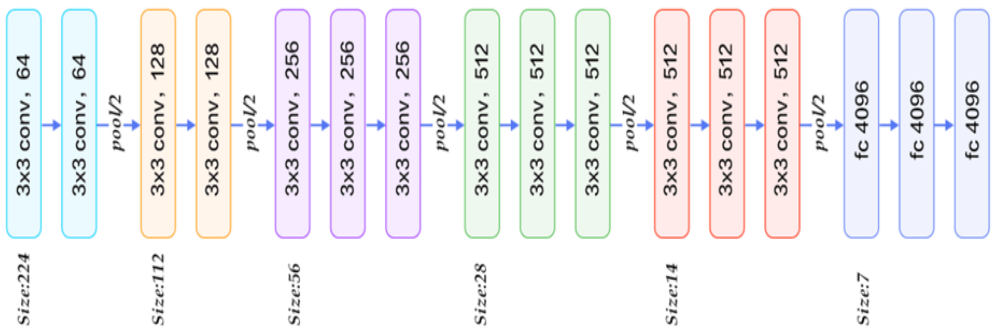
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) là một cuộc thi hàng năm của lĩnh vực thị giác máy tính. Mỗi năm, các đội cạnh tranh dựa trên 2 nhiệm vụ. Nhiệm vụ thứ nhất là nhận diện các đối tượng trong một hình ảnh với 200 class, được gọi là object localization. Nhiệm vụ thứ 2 là phân lớp ảnh, mỗi hình ảnh được gán nhãn một trong 1000 loại nhãn, được gọi là

Image classification. VGG-16 là mạng CNN được đề xuất bởi Karen Simonyan và Andrew Zisserman của Visual Goemetry Group Lab của trường đại học Oxford vào năm 2014 trong tờ báo có tiêu đề "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION". Model này đã đạt được hạng nhất và hạng hai trong các danh mục của cuộc thi ILSVRC năm 2014.



Cấu trúc VGG-16

Model sau khi train bởi mạng VGG-16 đạt độ chính xác 92.7% top-5 test trong bộ dữ liệu ImageNet gồm 14 triệu hình ảnh thuộc 1000 lớp khác nhau.



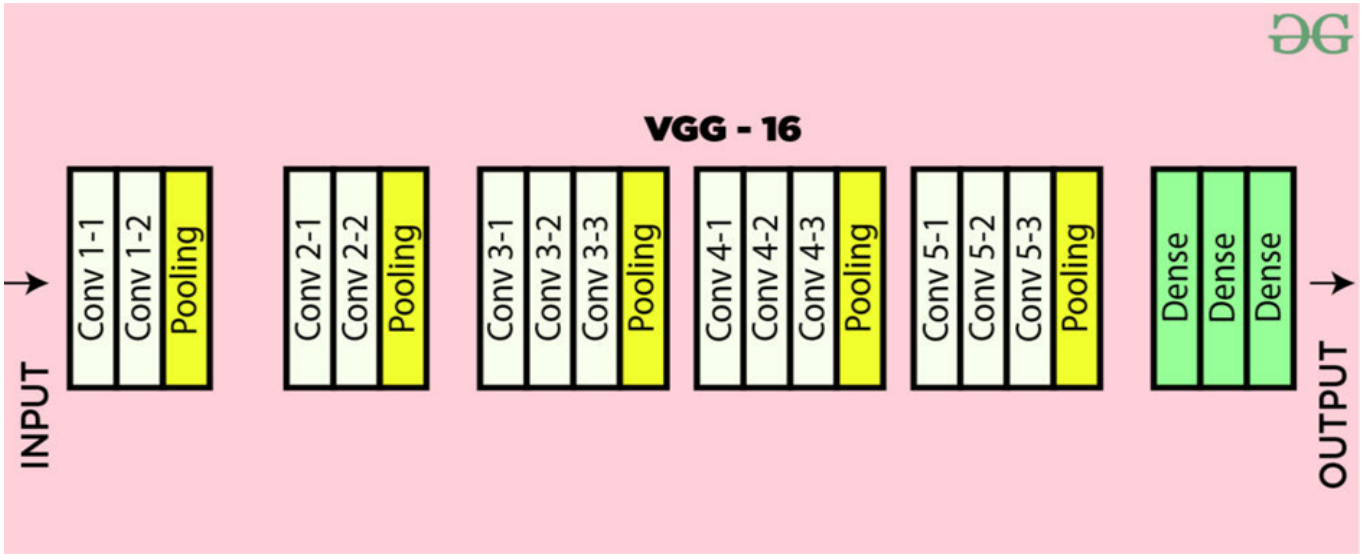
Cấu trúc VGG-16

conv: convolutional layer, pool: pooling layer, fc: fully connected layer

Phân tích:

- Convolutional layer: kích thước 3*3, padding=1, stride=1. Tại sao không ghi stride, padding mà vẫn biết? Vì mặc định sẽ là stride=1 và padding để cho output cùng width và height với input.
- Pool/2 : max pooling layer với size 2*2·
- 3*3 conv, 64: thì 64 là số kernel áp dụng trong layer đấy, hay depth của output của layer đấy.
- Càng các convolutional layer sau thì kích thước width, height càng giảm nhưng depth càng tăng.
- Sau khá nhiều convolutional layer và pooling layer thì dữ liệu được flatten và cho vào fully connected layer

Biểu đồ cấu trúc VGG-16



Cấu trúc VGG-16

- 16 trong VGG16 đề cập đến 16 lớp có trọng số. trong VGG16 có 13 convolutional layers, 5 max pooling layers và 3 dến layers, đúng ra nó có tất cả 21 lớp nhưng chỉ có 16 lớp trọng số, tức là lớp các tham số có thể học được.
- VGG16 lấy kích thước tensor đầu vào 224,224 với 3 RGB channel.
- Điều đặc biệt nhất của VGG16 là thay vì có một số lượng lớn các siêu tham số thì nó tập trung vào việc có các convolution layers của bộ lọc 3*3 với stride = 1 và luôn luôn sử dụng same padding và maxpool layer của bộ lọc 2*2 với stride = 2.
- Convolution và max pool layers được sắp xếp nhất quán trong toàn bộ cấu trúc.
- Conv-1 layer có 64 bộ lọc, Conv-2 có 128 bộ lọc, Conv-3 có 256 bộ lọc, Conv-4 và Còn-5 có 512 bộ lọc.
- 3 fully connected layer tuân theo một stack của convolutional layer: 2 lớp đầu tiên mỗi lớp có 4096 chanels, lớp thứ 3 thực hiện phân loại ILSVRC 1000 chiều vì vậy chứa 1000 chanels (một channel cho một class). Lớp cuối cùng là soft-max layer.

Cấu hình

Bảng bên dưới liệt kê các cấu trúc VGG khác nhau. Chúng ta có thể thấy rằng có 2 phiên bản VGG-16 (C và D). Không có nhiều điểm khác biệt giữa chúng ngoại trừ một số convolutional layer, convolutional sử dụng bộ lọc có kích thước (3,3) thay vì (1,1). Hai cái này chứa lần lượt 134 triệu và 138 triệu tham số tương ứng.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Các cấu hình VGG

Hạn chế của VGG-16

- Quá trình train rất chậm (model VGG ban đầu được train trên Nvidia Titan GPU trong 2-3 tuần).
- Kích thước file ImageNet Weight dùng để train cho VGG-16 có dung lượng lên đến 528MB. Vì vậy nó chiếm khá nhiều dung lượng ổ đĩa và băng thông khiến nó kém hiệu quả.
- 138 triệu tham số dẫn đến sự cố exploding gradients.

V. Kết quả

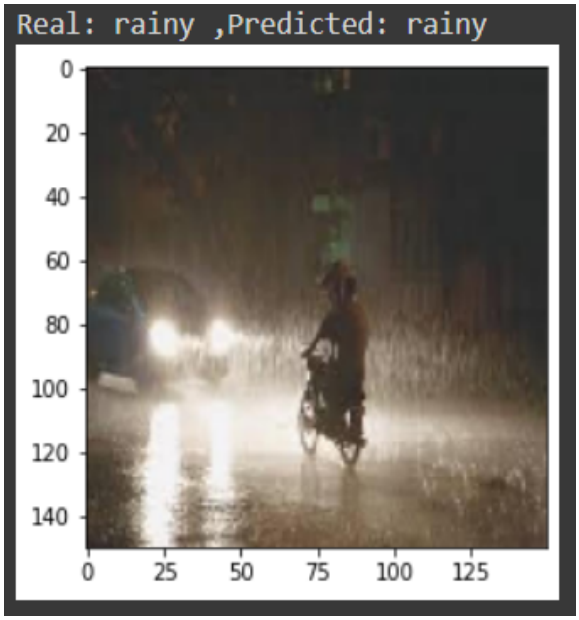
Cuối cùng ta thu được accuracy là 90% với bộ test dataset của model VGG16, còn với model 2 là 81.8% . Một accuracy cao và khá hiệu quả. Sau khi thu được accuracy cao thì ta sẽ sử dụng model VGG16 để dự đoán với một vài hình ảnh.

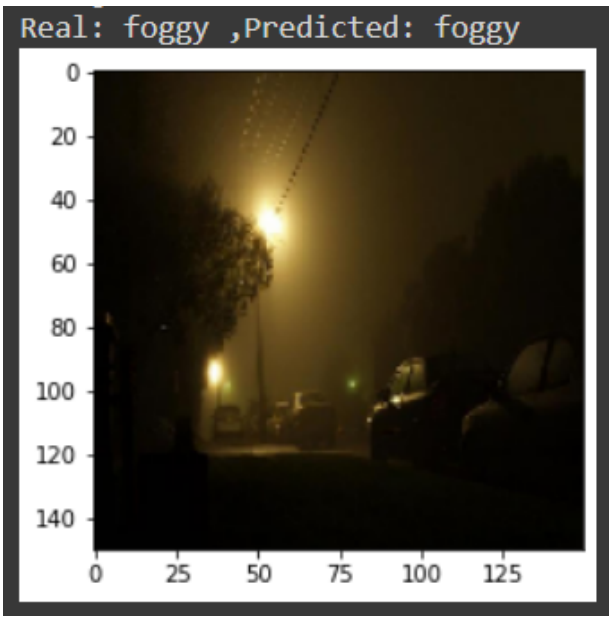
```
#model2
scores_tr = model2.evaluate(test_generator)
print('Test loss     :', scores_tr[0])
print('Test accuracy:', scores_tr[1])

#model with transfer learning
scores_tr = model_t.evaluate(test_generator)
print('Test loss     :', scores_tr[0])
print('Test accuracy:', scores_tr[1])
```

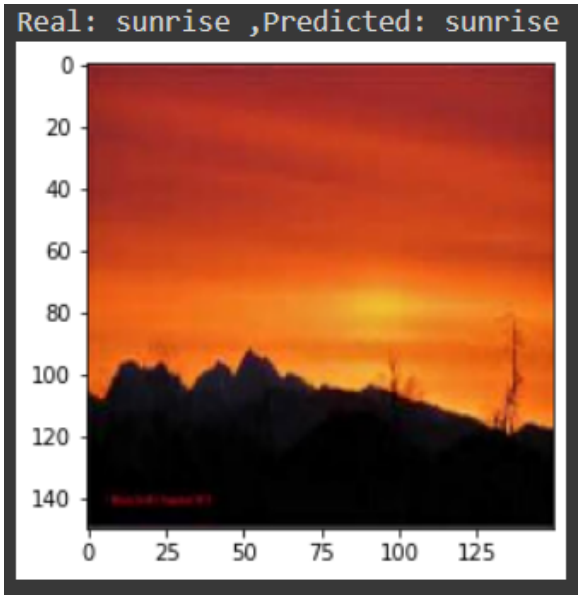
```
5/5 [==============================] - 1s 148ms/step -
Test loss     : 0.5621068477630615
Test accuracy: 0.8181818127632141
5/5 [==============================] - 1s 163ms/step -
Test loss     : 0.2713995575904846
Test accuracy: 0.9020978808403015
```

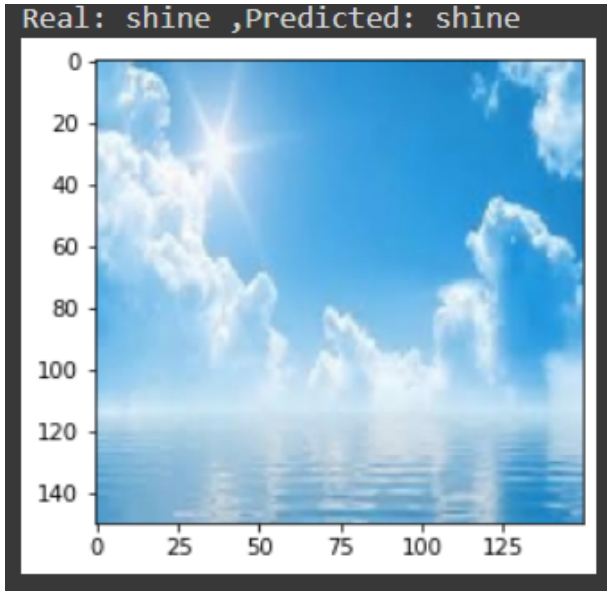
VI. Dự đoán

Trước hết là những dự đoán đúng từ model VGG16 mà nhóm đã dự đoán được.

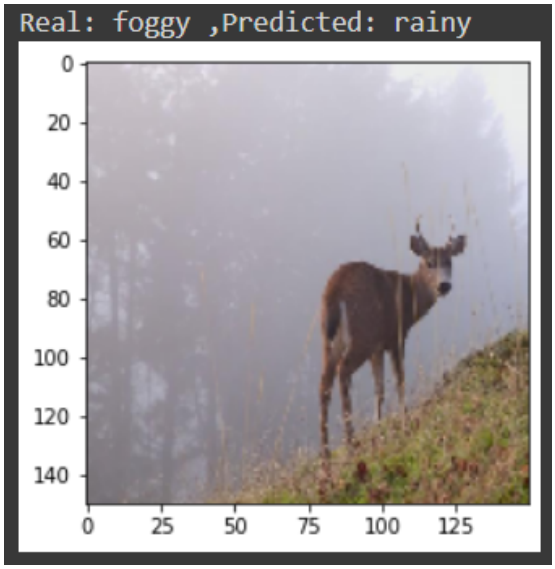


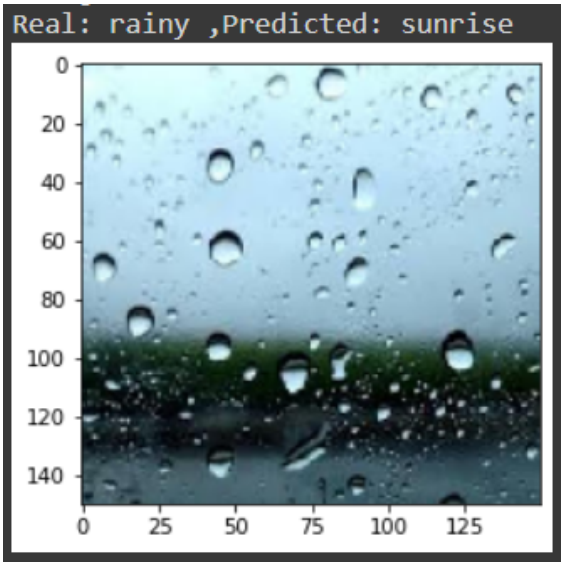






Bên cạnh đó cũng có một vài trường hợp dự đoán sai từ model VGG16:


Real: foggy ,Predicted: rainy
0
20
40
60
80
100
120
140
0
25
50
75
100
125


Real: rainy ,Predicted: sunrise
0
20
40
60
80
100
120
140
0
25
50
75
100
125

Tài liệu tham khảo:

https://wiki.tino.org/convolutional-neural-network-la-gi/
https://www.superdatascience.com/blogs/deep-learning-a-z-convolutional-neural-networks-cnn-step-3-flattening/
https://urvog.medium.com/weather-image-classification-with-keras-4eee9468ff2f
https://github.com/urvog/weatherclassification
https://www.geeksforgeeks.org/vgg-16-cnn-model/
https://phamdinhkhanh.github.io/2020/04/15/TransferLearning.html#:~:text=Transfer%20learning%20l%C3%A0%20m%E1%BB%99t%20trong,ta%20ph%E1%BA%A3i%20c%C3%B3%20kinh%20nghi%E1%BB%87m.