BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KĨ THUẬT TP.HCM
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

HCMUTE

BÁO CÁO

MÔN HỌC: TRÍ TUỆ NHÂN TẠO

Giảng viên hướng dẫn : PGS.TS Nguyễn Trường Thịnh
Sinh viên thực hiện : Nguyễn Tiến Đạt
MSSV : 20146322
Lớp : Chiều Thứ 6 – tiết 11-14
Mã Môn Học : ARIN337629_22_2_05CLC

TP.HCM,Ngày 21 Tháng 4 Năm 2023

BÀI TẬP SỐ 1 : NHẬN DIỆN 5 VÂN TAY VÀ DỰ ĐOÁN VỀ TƯƠNG LAI

1. GIỚI THIỆU

Vân tay là một đặc điểm sinh trắc học độc đáo của con người, được hình thành từ khi chúng ta còn ở trong bụng mẹ. Vân tay bao gồm các đường nét, vạch và hình ảnh trên bề mặt ngón tay và lòng bàn tay, có thể được sử dụng để xác định danh tính của một người.

Mỗi người có vân tay riêng biệt, không giống ai khác, và chúng không thay đổi suốt cuộc đời. Điều này làm cho vân tay trở thành một phương tiện rất quan trọng trong các lĩnh vực như pháp y, an ninh và tư pháp. Các chuyên gia có thể sử dụng kỹ thuật phân tích vân tay để giúp xác định tội phạm hoặc giải quyết các tranh chấp liên quan đến quyền sở hữu trí tuệ.

Ngoài ra, vân tay còn được sử dụng trong các lĩnh vực khác như công nghệ thông tin, điện tử, và thậm chí cả y học. Ví dụ, các điện thoại thông minh hiện nay có thể được mở khóa bằng vân tay và các nhà khoa học đã nghiên cứu sử dụng vân tay để chẩn đoán các bệnh lý của mắt.

2.PHƯƠNG PHÁP LUẬN

2.1 Thuật toán CNN

Thuật toán CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo được sử dụng trong bài toán phân loại ảnh và xử lý ảnh. Thuật toán này giúp cho việc nhận dạng, phân loại và xử lý ảnh trở nên chính xác hơn.

CNN bao gồm nhiều lớp, mỗi lớp đóng vai trò trong quá trình xử lý và trích xuất đặc trưng của ảnh. Bên trong mỗi lớp là một tập hợp các bộ lọc (filters), được áp dụng trên ảnh đầu vào để tạo ra các đặc trưng cụ thể của ảnh.

Các lớp trong CNN bao gồm:

Lớp Input: Lớp này chứa ảnh đầu vào và có kích thước tùy ý.

Lớp Convolution: Lớp này sử dụng bộ lọc để trích xuất đặc trưng của ảnh. Bộ lọc được áp dụng lên ảnh đầu vào và tạo ra các ma trận tích chập (convolution matrix). Các ma trận tích chập này có thể hiểu như là các bản đồ đặc trưng (feature maps) của ảnh đầu vào.

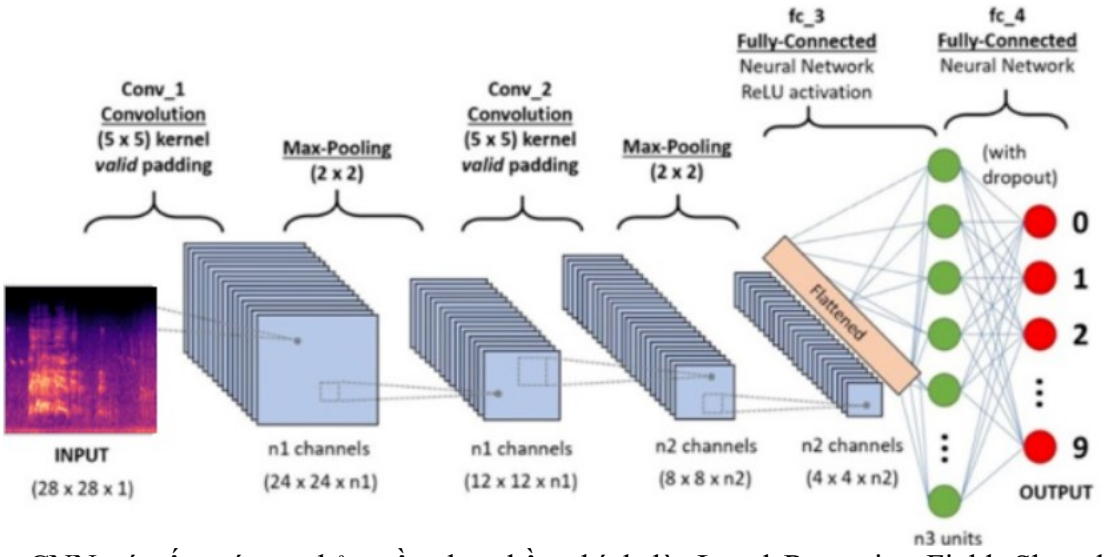
Lớp Activation: Lớp này áp dụng hàm kích hoạt (activation function) lên các bản đồ đặc trưng để tạo ra bản đồ đặc trưng kích hoạt (activated feature maps). Hàm kích hoạt thường được sử dụng là hàm ReLU (Rectified Linear Unit).

Lớp Pooling: Lớp này giảm kích thước của bản đồ đặc trưng bằng cách chọn ra các giá trị quan trọng nhất. Các phép toán thường được sử dụng trong lớp này là Max Pooling hoặc Average Pooling.

Lớp Fully Connected: Lớp này được sử dụng để kết nối các bản đồ đặc trưng đã được trích xuất từ các lớp trước đó và phân loại ảnh. Lớp này sử dụng một hoặc nhiều lớp fully connected để tính toán đầu ra cuối cùng.

CNN được sử dụng rộng rãi trong các bài toán liên quan đến xử lý ảnh và thị giác máy tính, chẳng hạn như nhận dạng khuôn mặt, phân loại chủ đề của ảnh, và phát hiện đối tượng trong ảnh.

2.2 Cấu trúc mạng CNN



CNN có cấu trúc cơ bản gồm ba phần chính là: Local Receptive Field, Shared Weights And Bias và Pooling.

Local Receptive Field (trường tiếp nhận cục bộ)

Đây được xem là lớp giúp bạn có thể tách lọc các dữ liệu, thông tin của ảnh và chọn được những vùng ảnh có giá trị sử dụng nhất.

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

Tuy nhiên trong CNN chúng ta không làm như vậy mà chúng ta chỉ kết nối trong một vùng nhỏ của các neuron đầu vào như một filter có kích thước 5×5 tương ứng (28- 5 + 1) 24 điểm ảnh đầu vào. Mỗi một kết nối sẽ học một trọng số và mỗi neuron ẩn sẽ học một bias. Mỗi một vùng 5×5 đấy gọi là một trường tiếp nhận cục bộ

Shared Weights And Bias (trọng số chia sẻ)

Chức năng chính của lớp này là hỗ trợ bạn làm giảm tối đa số lượng những tham số trong mạng CNN. Vì trong mỗi Convolution sẽ bao gồm các Feature Map khác nhau, mỗi Feature Map lại giúp Detect một vài Feature trong ảnh.

Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-rontrong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Tóm lại, một convolutional layer bao gồm các feature map khác nhau. Mỗi một feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN

Pooling Layer (lớp tổng hợp)

Đây được xem gần như là lớp cuối cùng trước khi đưa ra kết quả trong CNN. Chính vì thế, để có được kết quả dễ hiểu và dễ sử dụng nhất thì Pooling Layer có nhiệm vụ

làm đơn giản hóa các thông tin đầu ra. Nghĩa là, sau khi hoàn thành quá trình tính toán và quét các lớp thì sẽ đi đến Pooling Layer nhằm lượt bớt các không tin không cần thiết và cho ra kết quả mà chúng ta đang cần.

3.THỰC HIỆN

Bước 1: Khai Báo thư viện

```
#importing important libraries
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD, RMSprop,Adam
from tensorflow.keras.utils import to_categorical, load_img, img_to_array
from tensorflow.keras.models import load_model
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense,Flatten, Dropout, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
```

Bước 2: Liên kết Google drive

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Bước 3: Tăng cường dữ liệu đầu vào

```
train = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
train_data = '/content/drive/MyDrive/Colab Notebooks/Fingerprint/training_data'
validation = ImageDataGenerator(rescale=1./255)
valid_data = '/content/drive/MyDrive/Colab Notebooks/Fingerprint/training_data'
```

Bước 4: Xây dựng mô hình CNN

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same',input_shape=(256,256,3)))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(64,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())
model.add(Dense(128,activation='relu',kernel_initializer='he_uniform',))
model.add(Dense(5,activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 32)      896

 conv2d_1 (Conv2D)           (None, 256, 256, 32)      9248

 max_pooling2d (MaxPooling2D  (None, 128, 128, 32)     0
 )

 conv2d_2 (Conv2D)           (None, 128, 128, 32)      9248

 conv2d_3 (Conv2D)           (None, 128, 128, 32)      9248

 max_pooling2d_1 (MaxPooling  (None, 64, 64, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 64, 64, 64)        18496

 conv2d_5 (Conv2D)           (None, 64, 64, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 flatten (Flatten)           (None, 65536)             0

 dense (Dense)               (None, 128)               8388736

 dense_1 (Dense)             (None, 5)                 645

=================================================================
Total params: 8,473,445
Trainable params: 8,473,445
Non-trainable params: 0
```

Bước 5: Huấn luyện mô hình

```
from tensorflow.keras.optimizers import SGD
opt = SGD(lr=0.001, momentum=0.9) # lr learning rate : tốc độ học
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_dataset,epochs=10,batch_size=32,validation_data=validation_dataset,verbose=1)
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.SGD.
Epoch 1/10
53/53 [==============================] - 274s 5s/step - loss: 7.2252 - accuracy: 0.3055 - val_loss: 1.5653 - val_accuracy: 0.3359
Epoch 2/10
53/53 [==============================] - 285s 5s/step - loss: 1.6524 - accuracy: 0.3340 - val_loss: 1.5715 - val_accuracy: 0.3359
Epoch 3/10
53/53 [==============================] - 279s 5s/step - loss: 1.5872 - accuracy: 0.3131 - val_loss: 1.5480 - val_accuracy: 0.3359
Epoch 4/10
53/53 [==============================] - 285s 5s/step - loss: 1.5798 - accuracy: 0.3397 - val_loss: 1.5426 - val_accuracy: 0.3359
Epoch 5/10
53/53 [==============================] - 279s 5s/step - loss: 1.5665 - accuracy: 0.3245 - val_loss: 1.5424 - val_accuracy: 0.3359
Epoch 6/10
53/53 [==============================] - 280s 5s/step - loss: 1.5664 - accuracy: 0.3283 - val_loss: 1.5302 - val_accuracy: 0.3359
Epoch 7/10
53/53 [==============================] - 278s 5s/step - loss: 1.5667 - accuracy: 0.3378 - val_loss: 1.5688 - val_accuracy: 0.3359
Epoch 8/10
53/53 [==============================] - 240s 5s/step - loss: 1.5596 - accuracy: 0.3321 - val_loss: 1.5306 - val_accuracy: 0.3359
Epoch 9/10
53/53 [==============================] - 279s 5s/step - loss: 1.5436 - accuracy: 0.3321 - val_loss: 1.5204 - val_accuracy: 0.3359
Epoch 10/10
53/53 [==============================] - 279s 5s/step - loss: 1.5277 - accuracy: 0.3321 - val_loss: 1.5174 - val_accuracy: 0.3359
```

Bước 6 :Đánh giá độ chính xác

```
bai1_model = load_model('bai1.h5')
```

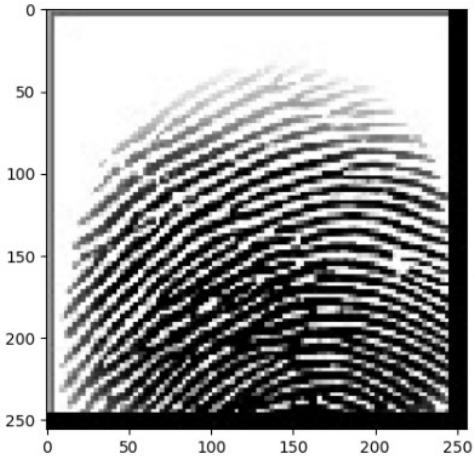
```
score=bai1_model.evaluate(validation_dataset,verbose=1)
print('Test loss = ',score[0])
print('Test accuracy = ',score[1])
```

```
53/53 [==============================] - 62s 1s/step - loss: 1.5174 - accuracy: 0.3359
Test loss =  1.517401933670044
Test accuracy =  0.3358633816242218
```

4.Kết Quả

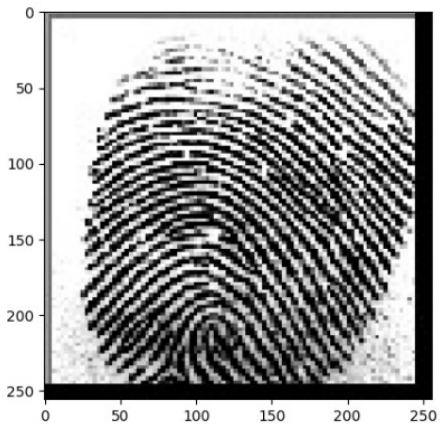
```
img = load_img('/content/drive/MyDrive/Colab Notebooks/Fingerprint/testing_data/Teacher (32).jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['Electrcian','Engineer','famer','teacher','worker']
print('Predict is: ',Label[int(np.argmax(bai1_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 89ms/step
Predict is:  worker
```



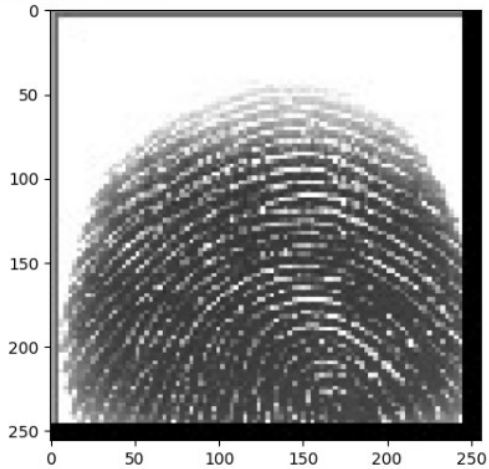
```
img = load_img('/content/drive/MyDrive/Colab Notebooks/Fingerprint/testing_data/Electrician (63).jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['Electrcian','Engineer','famer','teacher','worker']
print('Predict is: ',Label[int(np.argmax(bai1_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 421ms/step
Predict is:  worker
```




```
img = load_img('/content/drive/MyDrive/Colab Notebooks/Fingerprint/testing_data/Farmer (24).jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['Electrcian','Engineer','famer','teacher','worker']
print('Predict is: ',Label[int(np.argmax(bai1_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 166ms/step
Predict is:  worker
```



5.KẾT LUẬN

Thông qua Thuật toán CNN về bài tập nhận diện vân tay để dự đoán tương lai tỉ lệ chính xác khá thấp khoảng 40% không đáp ứng được mục tiêu của bài toán CNN cần phát triển thêm nhiều dữ liệu và số lần học

BÀI TẬP SỐ 2 : NHẬN DIỆN LOẠI TIỀN TỆ VIỆT NAM

1.GIỚI THIỆU

Tiền Việt Nam là đơn vị tiền tệ được sử dụng chính thức trong Việt Nam. Đồng tiền Việt Nam được phát hành bởi Ngân hàng Nhà nước Việt Nam và gồm nhiều loại giấy và kim loại với các mệnh giá khác nhau, bao gồm đồng xu và giấy tờ tiền.

Hiện tại, đồng xu Việt Nam có các mệnh giá là 200 đồng, 500 đồng và 1.000 đồng, trong khi giấy tờ tiền gồm các mệnh giá là 10.000 đồng, 20.000 đồng, 50.000 đồng, 100.000 đồng, 200.000 đồng và 500.000 đồng. Các loại giấy tờ tiền được thiết kế với các hình ảnh của các vị anh hùng lịch sử, danh lam thắng cảnh, động vật, cây cối và các biểu tượng của Việt Nam.

Trong quá khứ, tiền Việt Nam đã trải qua nhiều đổi mới và cải cách. Trước đây, tiền Việt Nam gồm hai đồng tiền riêng biệt là đồng Nam Việt và đồng Bắc Việt, trước khi được thống nhất vào một đồng tiền chung là đồng Việt Nam đồng. Sau đó, vào năm 1978, đồng Việt Nam đồng bị thay thế bằng đồng mới có tên gọi là đồng mới Việt Nam (đồng VND), và đồng này đã được sử dụng cho đến ngày nay.

2.PHƯƠNG PHÁP LUẬN

2.1 Thuật toán CNN

Thuật toán CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo được sử dụng trong bài toán phân loại ảnh và xử lý ảnh. Thuật toán này giúp cho việc nhận dạng, phân loại và xử lý ảnh trở nên chính xác hơn.

CNN bao gồm nhiều lớp, mỗi lớp đóng vai trò trong quá trình xử lý và trích xuất đặc trưng của ảnh. Bên trong mỗi lớp là một tập hợp các bộ lọc (filters), được áp dụng trên ảnh đầu vào để tạo ra các đặc trưng cụ thể của ảnh.

Các lớp trong CNN bao gồm:

Lớp Input: Lớp này chứa ảnh đầu vào và có kích thước tùy ý.

Lớp Convolution: Lớp này sử dụng bộ lọc để trích xuất đặc trưng của ảnh. Bộ lọc được áp dụng lên ảnh đầu vào và tạo ra các ma trận tích chập (convolution matrix). Các ma trận tích chập này có thể hiểu như là các bản đồ đặc trưng (feature maps) của ảnh đầu vào.

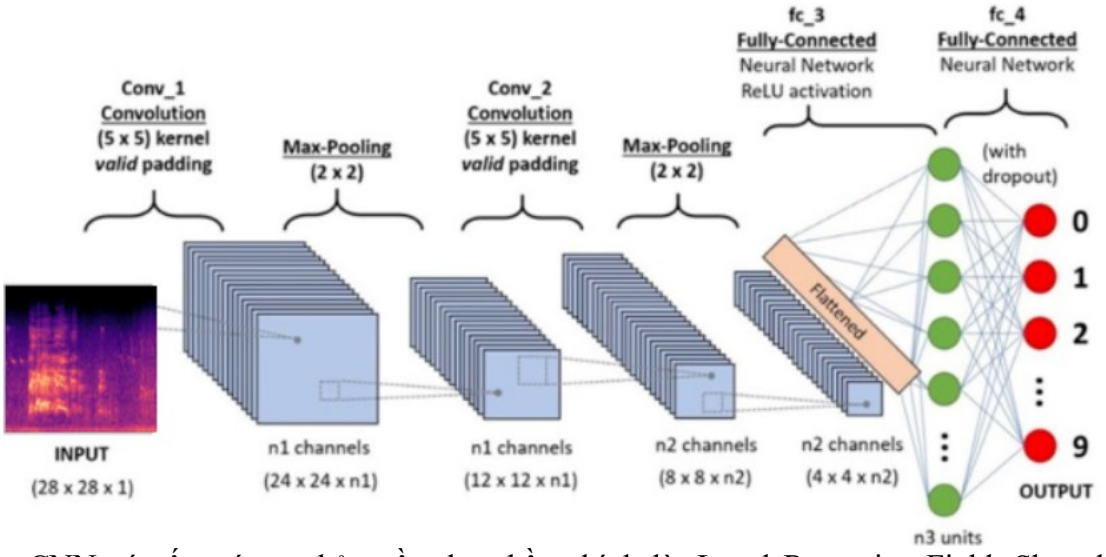
Lớp Activation: Lớp này áp dụng hàm kích hoạt (activation function) lên các bản đồ đặc trưng để tạo ra bản đồ đặc trưng kích hoạt (activated feature maps). Hàm kích hoạt thường được sử dụng là hàm ReLU (Rectified Linear Unit).

Lớp Pooling: Lớp này giảm kích thước của bản đồ đặc trưng bằng cách chọn ra các giá trị quan trọng nhất. Các phép toán thường được sử dụng trong lớp này là Max Pooling hoặc Average Pooling.

Lớp Fully Connected: Lớp này được sử dụng để kết nối các bản đồ đặc trưng đã được trích xuất từ các lớp trước đó và phân loại ảnh. Lớp này sử dụng một hoặc nhiều lớp fully connected để tính toán đầu ra cuối cùng.

CNN được sử dụng rộng rãi trong các bài toán liên quan đến xử lý ảnh và thị giác máy tính, chẳng hạn như nhận dạng khuôn mặt, phân loại chủ đề của ảnh, và phát hiện đối tượng trong ảnh.

2.2 Cấu trúc mạng CNN



CNN có cấu trúc cơ bản gồm ba phần chính là: Local Receptive Field, Shared Weights And Bias và Pooling.

Local Receptive Field (trường tiếp nhận cục bộ)

Đây được xem là lớp giúp bạn có thể tách lọc các dữ liệu, thông tin của ảnh và chọn được những vùng ảnh có giá trị sử dụng nhất.

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

Tuy nhiên trong CNN chúng ta không làm như vậy mà chúng ta chỉ kết nối trong một vùng nhỏ của các neuron đầu vào như một filter có kích thước 5×5 tương ứng (28- 5 + 1) 24 điểm ảnh đầu vào. Mỗi một kết nối sẽ học một trọng số và mỗi neuron ẩn sẽ học một bias. Mỗi một vùng 5×5 đấy gọi là một trường tiếp nhận cục bộ

Shared Weights And Bias (trọng số chia sẻ)

Chức năng chính của lớp này là hỗ trợ bạn làm giảm tối đa số lượng những tham số trong mạng CNN. Vì trong mỗi Convolution sẽ bao gồm các Feature Map khác nhau, mỗi Feature Map lại giúp Detect một vài Feature trong ảnh.

Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-rontrong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Tóm lại, một convolutional layer bao gồm các feature map khác nhau. Mỗi một feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN

Pooling Layer (lớp tổng hợp)

Đây được xem gần như là lớp cuối cùng trước khi đưa ra kết quả trong CNN. Chính vì thế, để có được kết quả dễ hiểu và dễ sử dụng nhất thì Pooling Layer có nhiệm vụ làm đơn giản hóa các thông tin đầu ra. Nghĩa là, sau khi hoàn thành quá trình tính toán và quét các lớp thì sẽ đi đến Pooling Layer nhằm lượt bớt các không tin không cần thiết và cho ra kết quả mà chúng ta đang cần.

3.THỰC HIỆN

Bước 1: Khai Báo thư viện

```
import numpy as np
from tensorflow import keras
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD,Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.utils import to_categorical
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
```

Bước 2: Liên kết Google drive

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

Bước 3: Tăng cường dữ liệu

```
train_datagen = ImageDataGenerator(rescale=1./255,validation_split = 0.2)
training_set=train_datagen.flow_from_directory('/content/drive/MyDrive/Colab Notebooks/Money/Money/training_data',
                                               target_size=(256,256),
                                               batch_size=32,
                                               class_mode ='categorical')
validation_set=train_datagen.flow_from_directory('/content/drive/MyDrive/Colab Notebooks/Money/Money/training_data',
                                               target_size=(256,256),
                                               batch_size=32,
                                               class_mode ='categorical')
```

```
Found 1179 images belonging to 11 classes.
Found 1179 images belonging to 11 classes.
```

Bước 4: Xây dựng mô hình CNN

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(256, 256,3)))
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
```

```
# example output part of the model
from keras.layers import Flatten
model.add(Flatten())
model.add(Dense(256, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(11, activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 256, 256, 32)      896

 conv2d_5 (Conv2D)           (None, 256, 256, 32)      9248

 max_pooling2d_2 (MaxPooling  (None, 128, 128, 32)     0
 2D)

 conv2d_6 (Conv2D)           (None, 128, 128, 64)      18496

 conv2d_7 (Conv2D)           (None, 128, 128, 64)      36928

 max_pooling2d_3 (MaxPooling  (None, 64, 64, 64)       0
 2D)

 conv2d_8 (Conv2D)           (None, 64, 64, 128)       73856

 conv2d_9 (Conv2D)           (None, 64, 64, 128)       147584

 max_pooling2d_4 (MaxPooling  (None, 32, 32, 128)      0
 2D)

 flatten_1 (Flatten)         (None, 131072)            0

 dense_2 (Dense)             (None, 256)               33554688

 dense_3 (Dense)             (None, 11)                2827

=================================================================
Total params: 33,844,523
Trainable params: 33,844,523
Non-trainable params: 0
```

Bước 5: Huấn luyện mô hình

```
from tensorflow.keras.optimizers import SGD
opt = SGD(lr=0.001, momentum=0.9) # lr learning rate : tốc độ học
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(training_set,epochs=20,batch_size=64,validation_data=validation_set,verbose=1)
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.SGD.
Epoch 1/20
37/37 [==============================] - 13s 289ms/step - loss: 9.4104 - accuracy: 0.6404 - val_loss: 0.4287 - val_accuracy: 0.8991
Epoch 2/20
37/37 [==============================] - 11s 297ms/step - loss: 0.1424 - accuracy: 0.9729 - val_loss: 4.8918e-04 - val_accuracy: 1.0000
Epoch 3/20
37/37 [==============================] - 10s 282ms/step - loss: 1.5903e-04 - accuracy: 1.0000 - val_loss: 3.0870e-05 - val_accuracy: 1.0000
Epoch 4/20
37/37 [==============================] - 11s 285ms/step - loss: 2.2974e-05 - accuracy: 1.0000 - val_loss: 8.4753e-06 - val_accuracy: 1.0000
Epoch 5/20
37/37 [==============================] - 10s 278ms/step - loss: 3.8630e-06 - accuracy: 1.0000 - val_loss: 1.8129e-06 - val_accuracy: 1.0000
Epoch 6/20
37/37 [==============================] - 11s 291ms/step - loss: 1.2553e-06 - accuracy: 1.0000 - val_loss: 7.4882e-07 - val_accuracy: 1.0000
Epoch 7/20
37/37 [==============================] - 11s 292ms/step - loss: 5.8967e-07 - accuracy: 1.0000 - val_loss: 4.3538e-07 - val_accuracy: 1.0000
Epoch 8/20
37/37 [==============================] - 13s 357ms/step - loss: 3.7229e-07 - accuracy: 1.0000 - val_loss: 3.1132e-07 - val_accuracy: 1.0000
Epoch 9/20
37/37 [==============================] - 11s 296ms/step - loss: 2.7542e-07 - accuracy: 1.0000 - val_loss: 2.3811e-07 - val_accuracy: 1.0000
Epoch 10/20
37/37 [==============================] - 11s 304ms/step - loss: 2.2143e-07 - accuracy: 1.0000 - val_loss: 1.9767e-07 - val_accuracy: 1.0000
Epoch 11/20
37/37 [==============================] - 11s 297ms/step - loss: 1.8584e-07 - accuracy: 1.0000 - val_loss: 1.7037e-07 - val_accuracy: 1.0000
Epoch 12/20
37/37 [==============================] - 11s 307ms/step - loss: 1.5955e-07 - accuracy: 1.0000 - val_loss: 1.4509e-07 - val_accuracy: 1.0000
Epoch 13/20
37/37 [==============================] - 10s 276ms/step - loss: 1.4064e-07 - accuracy: 1.0000 - val_loss: 1.3094e-07 - val_accuracy: 1.0000
Epoch 14/20
37/37 [==============================] - 11s 296ms/step - loss: 1.2578e-07 - accuracy: 1.0000 - val_loss: 1.1668e-07 - val_accuracy: 1.0000
Epoch 15/20
37/37 [==============================] - 11s 293ms/step - loss: 1.1355e-07 - accuracy: 1.0000 - val_loss: 1.0799e-07 - val_accuracy: 1.0000
Epoch 16/20
37/37 [==============================] - 11s 297ms/step - loss: 1.0364e-07 - accuracy: 1.0000 - val_loss: 9.9088e-08 - val_accuracy: 1.0000
Epoch 17/20
37/37 [==============================] - 11s 289ms/step - loss: 9.5549e-08 - accuracy: 1.0000 - val_loss: 9.0898e-08 - val_accuracy: 1.0000
Epoch 18/20
37/37 [==============================] - 11s 302ms/step - loss: 8.8472e-08 - accuracy: 1.0000 - val_loss: 8.5135e-08 - val_accuracy: 1.0000
Epoch 19/20
37/37 [==============================] - 11s 294ms/step - loss: 8.3012e-08 - accuracy: 1.0000 - val_loss: 7.9068e-08 - val_accuracy: 1.0000
Epoch 20/20
37/37 [==============================] - 11s 309ms/step - loss: 7.6945e-08 - accuracy: 1.0000 - val_loss: 7.3406e-08 - val_accuracy: 1.0000
```

Bước 6 :Đánh giá độ chính xác

```
score=tien10_model.evaluate(validation_set,verbose=1)
print('Test loss = ',score[0])
print('Test accuracy = ',score[1])
```

```
37/37 [==============================] - 5s 133ms/step - loss: 1.4883e-07 - accuracy: 1.0000
Test loss =  1.4883457311043458e-07
Test accuracy =  1.0
```

4.KẾT QUẢ

```
img = load_img('/content/drive/MyDrive/Colab Notebooks/Money/Money/testing_data/500k.jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['100k',
 '10k',
 '1k',
 '200k',
 '2k',
 '500k',
 '5k']
print('Predict is: ',Label[int(np.argmax(classificationFood_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 20ms/step
Predict is:  10k
```


```
img = load_img('/content/drive/MyDrive/Colab Notebooks/Money/Money/testing_data/100k.jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['100k',
 '10k',
 '1k',
 '200k',
 '2k',
 '500k',
 '5k']
print('Predict is: ',Label[int(np.argmax(classificationFood_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 348ms/step
Predict is:  100k
```

```
img = load_img('/content/drive/MyDrive/Colab Notebooks/Money/Money/testing_data/200k.jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['100k',
 '10k',
 '1k',
 '200k',
 '2k',
 '500k',
 '5k']
print('Predict is: ',Label[int(np.argmax(classificationFood_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 27ms/step
Predict is:  10k
```

5.KẾT LUẬN

Thuật Toán CNN có độ chính xác về nhận diện cao đối với bài tiền tệ tỉ lệ chính xác là 90% và cho ra kết quả chính xác , nếu như dữ liệu càng nhiều thì tỉ lệ chính xác càng cao

BÀI TẬP SỐ 3 : NHẬN DIỆN 10 MÓN ĂN CỦA VIỆT NAM

1.GIỚI THIỆU

Món ăn Việt Nam là một trong những đặc sản ẩm thực phổ biến và được yêu thích trên khắp thế giới. Với sự phối hợp đồng điệu của các vị gia vị tự nhiên, giá trị dinh dưỡng cao và việc sử dụng nguyên liệu tươi ngon, món ăn Việt Nam đem lại cho thực khách không chỉ món ăn ngon miệng mà còn là trải nghiệm văn hóa độc đáo.

Với những văn hóa độc đáo của từng vùng miền trên khắp đất nước hình chữ s này và với sự đa dạng của nguyên liệu và cách chế biến những món ăn theo địa phương ở Việt Nam khi người ta nhắc đến như Bánh mì việt Nam, Phở Hà Nội,… Cho thấy sư đa dạng về hương vị của món ăn mà còn là cách ăn và thưởng thức. Tất cả đều tạo nên một nền văn hoá ẩm thực đặc trưng và độc đáo của đất nước Việt Nam.

Với bài báo cáo này em dùng *mạng nơ-ron tích chập (cnn)* để thông qua các Tập dữ liệu hình ảnh được tìm kiếm trên internet để huấn luyên mô hình nhận các món ăn

2. PHƯƠNG PHÁP LUẬN

2.1 Thuật toán CNN

Thuật toán CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo được sử dụng trong bài toán phân loại ảnh và xử lý ảnh. Thuật toán này giúp cho việc nhận dạng, phân loại và xử lý ảnh trở nên chính xác hơn.

CNN bao gồm nhiều lớp, mỗi lớp đóng vai trò trong quá trình xử lý và trích xuất đặc trưng của ảnh. Bên trong mỗi lớp là một tập hợp các bộ lọc (filters), được áp dụng trên ảnh đầu vào để tạo ra các đặc trưng cụ thể của ảnh.

Các lớp trong CNN bao gồm:

Lớp Input: Lớp này chứa ảnh đầu vào và có kích thước tùy ý.

Lớp Convolution: Lớp này sử dụng bộ lọc để trích xuất đặc trưng của ảnh. Bộ lọc được áp dụng lên ảnh đầu vào và tạo ra các ma trận tích chập (convolution matrix). Các ma trận tích chập này có thể hiểu như là các bản đồ đặc trưng (feature maps) của ảnh đầu vào.

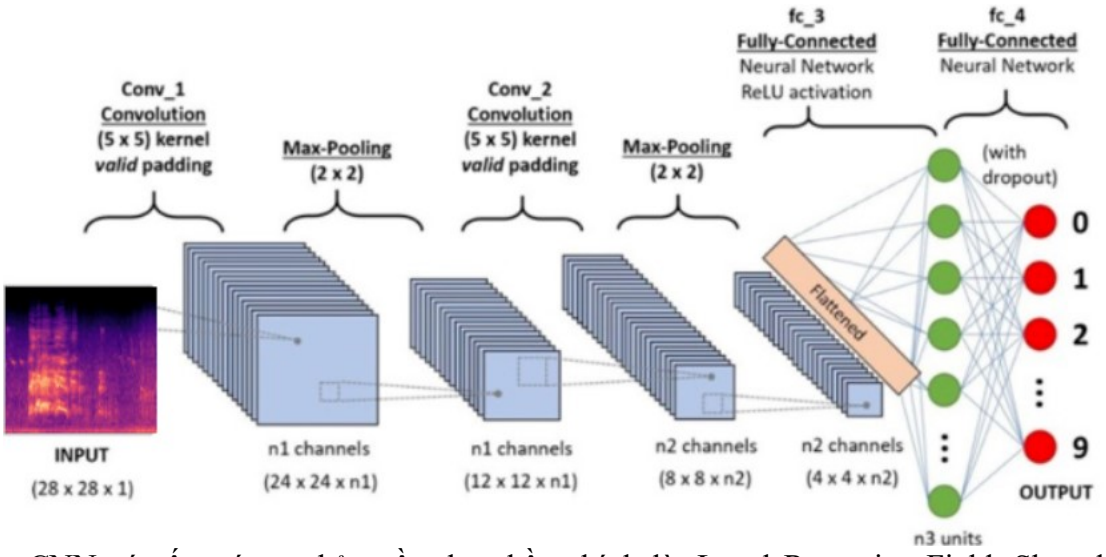
Lớp Activation: Lớp này áp dụng hàm kích hoạt (activation function) lên các bản đồ đặc trưng để tạo ra bản đồ đặc trưng kích hoạt (activated feature maps). Hàm kích hoạt thường được sử dụng là hàm ReLU (Rectified Linear Unit).

Lớp Pooling: Lớp này giảm kích thước của bản đồ đặc trưng bằng cách chọn ra các giá trị quan trọng nhất. Các phép toán thường được sử dụng trong lớp này là Max Pooling hoặc Average Pooling.

Lớp Fully Connected: Lớp này được sử dụng để kết nối các bản đồ đặc trưng đã được trích xuất từ các lớp trước đó và phân loại ảnh. Lớp này sử dụng một hoặc nhiều lớp fully connected để tính toán đầu ra cuối cùng.

CNN được sử dụng rộng rãi trong các bài toán liên quan đến xử lý ảnh và thị giác máy tính, chẳng hạn như nhận dạng khuôn mặt, phân loại chủ đề của ảnh, và phát hiện đối tượng trong ảnh.

2.2 Cấu trúc mạng CNN



CNN có cấu trúc cơ bản gồm ba phần chính là: Local Receptive Field, Shared Weights And Bias và Pooling.

Local Receptive Field (trường tiếp nhận cục bộ)

Đây được xem là lớp giúp bạn có thể tách lọc các dữ liệu, thông tin của ảnh và chọn được những vùng ảnh có giá trị sử dụng nhất.

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

Tuy nhiên trong CNN chúng ta không làm như vậy mà chúng ta chỉ kết nối trong một vùng nhỏ của các neuron đầu vào như một filter có kích thước 5×5 tương ứng (28- 5 + 1) 24 điểm ảnh đầu vào. Mỗi một kết nối sẽ học một trọng số và mỗi neuron ẩn sẽ học một bias. Mỗi một vùng 5×5 đấy gọi là một trường tiếp nhận cục bộ

Shared Weights And Bias (trọng số chia sẻ)

Chức năng chính của lớp này là hỗ trợ bạn làm giảm tối đa số lượng những tham số trong mạng CNN. Vì trong mỗi Convolution sẽ bao gồm các Feature Map khác nhau, mỗi Feature Map lại giúp Detect một vài Feature trong ảnh.

Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-rontrong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Tóm lại, một convolutional layer bao gồm các feature map khác nhau. Mỗi một feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN

Pooling Layer (lớp tổng hợp)

Đây được xem gần như là lớp cuối cùng trước khi đưa ra kết quả trong CNN. Chính vì thế, để có được kết quả dễ hiểu và dễ sử dụng nhất thì Pooling Layer có nhiệm vụ

làm đơn giản hóa các thông tin đầu ra. Nghĩa là, sau khi hoàn thành quá trình tính toán và quét các lớp thì sẽ đi đến Pooling Layer nhằm lượt bớt các không tin không cần thiết và cho ra kết quả mà chúng ta đang cần.

3.THỰC HIỆN

Bước 1: Khai Báo thư viện

```
#importing important libraries
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD, RMSprop,Adam
from tensorflow.keras.utils import to_categorical, load_img, img_to_array
from tensorflow.keras.models import load_model
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense,Flatten, Dropout, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
```

Bước 2: Liên kết Google drive

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

```
Mounted at /content/drive
```

Bước 3: Tăng cường dữ liệu đầu vào

```
train = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
train_data = '/content/drive/MyDrive/Colab Notebooks/monan10/training_data'
validation = ImageDataGenerator(rescale=1./255)
valid_data = '/content/drive/MyDrive/Colab Notebooks/monan10/training_data'
```

```
train_dataset = train.flow_from_directory(train_data, target_size = (150,150), batch_size = 10, class_mode = 'categorical')
validation_dataset = validation.flow_from_directory(valid_data, target_size = (150,150), batch_size = 10, class_mode = 'categorical')
```

```
Found 63 images belonging to 10 classes.
Found 63 images belonging to 10 classes.
```

Bước 4: Xây dựng mô hình CNN

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same',input_shape=(150,150,3)))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(64,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(128,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(128,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(256,activation='relu',kernel_initializer='he_uniform',))
model.add(Dense(10,activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      896

 conv2d_1 (Conv2D)           (None, 150, 150, 32)      9248

 max_pooling2d (MaxPooling2D  (None, 75, 75, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 75, 75, 32)        9248

 conv2d_3 (Conv2D)           (None, 75, 75, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 37, 37, 64)        18496

 conv2d_5 (Conv2D)           (None, 37, 37, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 18, 18, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 18, 18, 128)       73856

 conv2d_7 (Conv2D)           (None, 18, 18, 128)       147584

 max_pooling2d_3 (MaxPooling  (None, 9, 9, 128)        0
 2D)

 flatten (Flatten)           (None, 10368)             0

 dense (Dense)               (None, 256)               2654464

 dense_1 (Dense)             (None, 10)                2570

=================================================================
Total params: 2,962,538
Trainable params: 2,962,538
Non-trainable params: 0
```

Bước 5: Huấn luyện mô hình

```
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics = ['accuracy'])
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(SGD, self).__init__(name, **kwargs)
```

```
history = model.fit(train_dataset,batch_size=32,epochs=50,verbose=1,validation_data=validation_dataset)
```

```
Epoch 22/50
7/7 [==============================] - 1s 148ms/step - loss: 0.9247 - accuracy: 0.6825 - val_loss: 0.4412 - val_accuracy: 0.8571
Epoch 23/50
7/7 [==============================] - 1s 142ms/step - loss: 0.5773 - accuracy: 0.7619 - val_loss: 0.3888 - val_accuracy: 0.8730
Epoch 24/50
7/7 [==============================] - 1s 136ms/step - loss: 0.5119 - accuracy: 0.7937 - val_loss: 0.3234 - val_accuracy: 0.9048
Epoch 25/50
7/7 [==============================] - 1s 137ms/step - loss: 0.4042 - accuracy: 0.8571 - val_loss: 0.1934 - val_accuracy: 0.9524
Epoch 26/50
7/7 [==============================] - 1s 136ms/step - loss: 0.2421 - accuracy: 0.9206 - val_loss: 0.2343 - val_accuracy: 0.9206
Epoch 27/50
7/7 [==============================] - 1s 133ms/step - loss: 0.4010 - accuracy: 0.8571 - val_loss: 0.1433 - val_accuracy: 0.9683
Epoch 28/50
7/7 [==============================] - 1s 134ms/step - loss: 0.3449 - accuracy: 0.9048 - val_loss: 0.0919 - val_accuracy: 0.9841
Epoch 29/50
7/7 [==============================] - 1s 135ms/step - loss: 0.2859 - accuracy: 0.9048 - val_loss: 0.4253 - val_accuracy: 0.8571
Epoch 30/50
7/7 [==============================] - 1s 136ms/step - loss: 0.3163 - accuracy: 0.9048 - val_loss: 0.5535 - val_accuracy: 0.8413
Epoch 31/50
7/7 [==============================] - 1s 145ms/step - loss: 0.4660 - accuracy: 0.8413 - val_loss: 0.6067 - val_accuracy: 0.7302
Epoch 32/50
7/7 [==============================] - 1s 141ms/step - loss: 0.5987 - accuracy: 0.7778 - val_loss: 0.6932 - val_accuracy: 0.7937
Epoch 33/50
7/7 [==============================] - 1s 135ms/step - loss: 0.5063 - accuracy: 0.7937 - val_loss: 0.3855 - val_accuracy: 0.8571
Epoch 34/50
7/7 [==============================] - 1s 154ms/step - loss: 0.8007 - accuracy: 0.7460 - val_loss: 0.4742 - val_accuracy: 0.8889
Epoch 35/50
7/7 [==============================] - 1s 139ms/step - loss: 0.5077 - accuracy: 0.8254 - val_loss: 0.1434 - val_accuracy: 0.9841
Epoch 36/50
7/7 [==============================] - 1s 141ms/step - loss: 0.1872 - accuracy: 0.9683 - val_loss: 0.0924 - val_accuracy: 1.0000
Epoch 37/50
7/7 [==============================] - 1s 155ms/step - loss: 0.2178 - accuracy: 0.9206 - val_loss: 0.0477 - val_accuracy: 0.9841
Epoch 38/50
7/7 [==============================] - 1s 136ms/step - loss: 0.1355 - accuracy: 0.9365 - val_loss: 0.0803 - val_accuracy: 0.9683
Epoch 39/50
7/7 [==============================] - 1s 133ms/step - loss: 0.4257 - accuracy: 0.8730 - val_loss: 0.1357 - val_accuracy: 0.9841
Epoch 40/50
```

Bước 6 :Đánh giá độ chính xác

```
monan10_model = load_model('monan10.h5')
```

```
score=monan10_model.evaluate(validation_dataset,verbose=1)
print('Test loss = ',score[0])
print('Test accuracy = ',score[1])
```

```
7/7 [==============================] - 0s 42ms/step - loss: 0.0095 - accuracy: 1.0000
Test loss =  0.009476926177740097
Test accuracy =  1.0
```

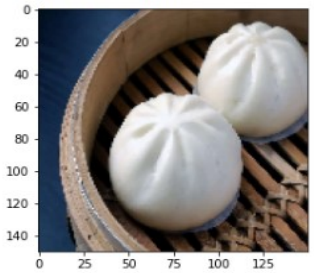
4.KẾT QUẢ

```
img = load_img('/content/drive/MyDrive/monantest/Pred_Data/bokho.jpg',target_size=(150,150))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255
Label = ['Banhbao',
 'Banhcuon',
 'Banhgai',
 'Banhmi',
 'Banhtrangcuon',
 'Bokho',
 'Bunbo',
 'Chagio',
 'Comtam',
 'Hutieu']
print('Predict is: ',Label[int(np.argmax(monan10_model.predict(img),axis=-1))])
```

```
Predict is:  Bokho
```

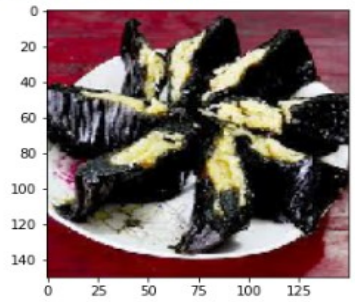

```
img = load_img('/content/drive/MyDrive/monantest/Pred_Data/2.jpg',target_size=(150,150))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255
Label = ['Banhbao',
 'Banhcuon',
 'Banhgai',
 'Banhmi',
 'Banhtrangcuon',
 'Bokho',
 'Bunbo',
 'Chagio',
 'Comtam',
 'Hutieu']
print('Predict is: ',Label[int(np.argmax(monan10_model.predict(img),axis=-1))])
```

```
Predict is:  Banhbao
```



```
img = load_img('/content/drive/MyDrive/monan10/Pred_Data/banhgai.jpg',target_size=(150,150))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255
Label = ['Banhbao',
 'Banhcuon',
 'Banhgai',
 'Banhmi',
 'Banhtrangcuon',
 'Bokho',
 'Bunbo',
 'Chagio',
 'Comtam',
 'Hutieu']
print('Predict is: ',Label[int(np.argmax(monan10d_model.predict(img),axis=-1))])
```

```
Predict is:  Banhgai
```



5.KẾT LUẬN

Thông qua thuật toán CNN và với bài tập về nhận diện món ăn của Việt Nam tỉ lệ chính xác cao , tỉ lệ chính xác 100% với số lần học 50 lần . do vậy thuật toán CNN đã đáp ứng được nhu cầu nhưng để có được hiệu quả thì cần thêm nhiều tập dữ liệu.

BÀI TẬP SỐ 4 : NHẬN DIỆN 5 LOÀI HOA

1.GIỚI THIỆU

Loài hoa là một loại thực vật có hoa và được phân loại vào ngành thực vật có hoa (Angiospermae). Loài hoa được chia thành nhiều họ, chi và loài khác nhau, với khoảng 300.000 loài đã được mô tả.

Mỗi loài hoa có các đặc điểm khác nhau, bao gồm kích thước, màu sắc, hình dạng, mùi hương, thời gian nở hoa và phương pháp thụ phấn. Các loài hoa có thể được phân loại theo màu sắc, như hoa đỏ, hoa vàng, hoa xanh lá cây, hoa tím, hoa trắng và hoa cam. Một số loài hoa phổ biến bao gồm hoa hồng, hoa cúc, hoa lan, hoa ly, hoa đỗ quyên và hoa tuyết tùng.

Các loài hoa thường được sử dụng cho mục đích trang trí, tôn giáo, y học, chế biến thực phẩm và nhiều mục đích khác. Ngoài ra, các loài hoa còn có vai trò quan trọng trong sinh thái học, bởi vì chúng cung cấp thức ăn cho động vật, giúp trong quá trình thụ phấn và phát triển của các loài thực vật khác.

2.PHƯƠNG PHÁP LUẬN

2.1 Thuật toán CNN

Thuật toán CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo được sử dụng trong bài toán phân loại ảnh và xử lý ảnh. Thuật toán này giúp cho việc nhận dạng, phân loại và xử lý ảnh trở nên chính xác hơn.

CNN bao gồm nhiều lớp, mỗi lớp đóng vai trò trong quá trình xử lý và trích xuất đặc trưng của ảnh. Bên trong mỗi lớp là một tập hợp các bộ lọc (filters), được áp dụng trên ảnh đầu vào để tạo ra các đặc trưng cụ thể của ảnh.

Các lớp trong CNN bao gồm:

Lớp Input: Lớp này chứa ảnh đầu vào và có kích thước tùy ý.

Lớp Convolution: Lớp này sử dụng bộ lọc để trích xuất đặc trưng của ảnh. Bộ lọc được áp dụng lên ảnh đầu vào và tạo ra các ma trận tích chập (convolution matrix). Các ma trận tích chập này có thể hiểu như là các bản đồ đặc trưng (feature maps) của ảnh đầu vào.

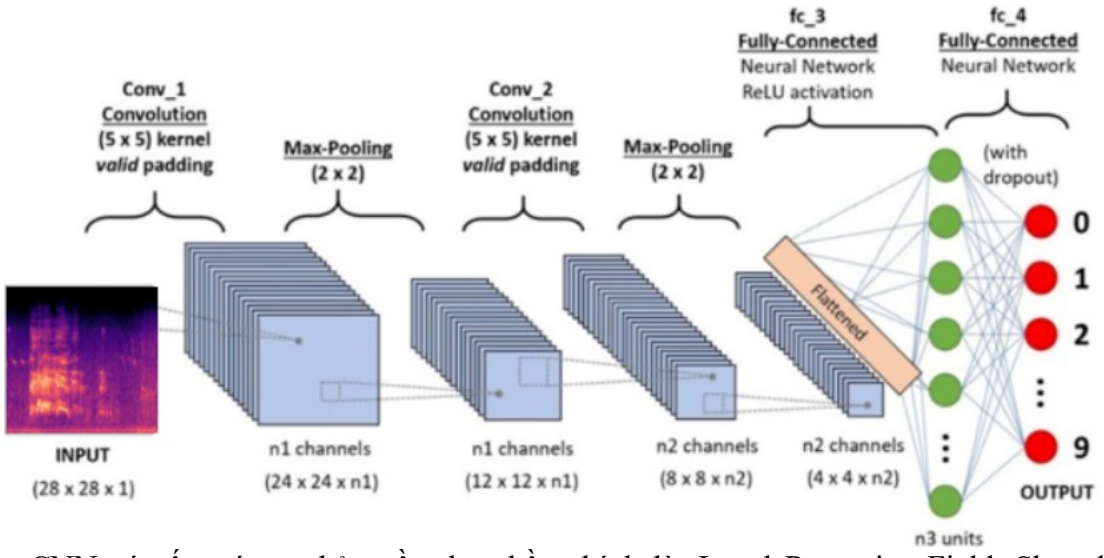
Lớp Activation: Lớp này áp dụng hàm kích hoạt (activation function) lên các bản đồ đặc trưng để tạo ra bản đồ đặc trưng kích hoạt (activated feature maps). Hàm kích hoạt thường được sử dụng là hàm ReLU (Rectified Linear Unit).

Lớp Pooling: Lớp này giảm kích thước của bản đồ đặc trưng bằng cách chọn ra các giá trị quan trọng nhất. Các phép toán thường được sử dụng trong lớp này là Max Pooling hoặc Average Pooling.

Lớp Fully Connected: Lớp này được sử dụng để kết nối các bản đồ đặc trưng đã được trích xuất từ các lớp trước đó và phân loại ảnh. Lớp này sử dụng một hoặc nhiều lớp fully connected để tính toán đầu ra cuối cùng.

CNN được sử dụng rộng rãi trong các bài toán liên quan đến xử lý ảnh và thị giác máy tính, chẳng hạn như nhận dạng khuôn mặt, phân loại chủ đề của ảnh, và phát hiện đối tượng trong ảnh.

2.2 Cấu trúc mạng CNN



CNN có cấu trúc cơ bản gồm ba phần chính là: Local Receptive Field, Shared Weights And Bias và Pooling.

Local Receptive Field (trường tiếp nhận cục bộ)

Đây được xem là lớp giúp bạn có thể tách lọc các dữ liệu, thông tin của ảnh và chọn được những vùng ảnh có giá trị sử dụng nhất.

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

Tuy nhiên trong CNN chúng ta không làm như vậy mà chúng ta chỉ kết nối trong một vùng nhỏ của các neuron đầu vào như một filter có kích thước 5×5 tương ứng (28- 5 + 1) 24 điểm ảnh đầu vào. Mỗi một kết nối sẽ học một trọng số và mỗi neuron ẩn sẽ học một bias. Mỗi một vùng 5×5 đấy gọi là một trường tiếp nhận cục bộ

Shared Weights And Bias (trọng số chia sẻ)

Chức năng chính của lớp này là hỗ trợ bạn làm giảm tối đa số lượng những tham số trong mạng CNN. Vì trong mỗi Convolution sẽ bao gồm các Feature Map khác nhau, mỗi Feature Map lại giúp Detect một vài Feature trong ảnh.

Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-rontrong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Tóm lại, một convolutional layer bao gồm các feature map khác nhau. Mỗi một feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN

Pooling Layer (lớp tổng hợp)

Đây được xem gần như là lớp cuối cùng trước khi đưa ra kết quả trong CNN. Chính vì thế, để có được kết quả dễ hiểu và dễ sử dụng nhất thì Pooling Layer có nhiệm vụ làm đơn giản hóa các thông tin đầu ra. Nghĩa là, sau khi hoàn thành quá trình tính toán và quét các lớp thì sẽ đi đến Pooling Layer nhằm lượt bớt các không tin không cần thiết và cho ra kết quả mà chúng ta đang cần.

3.THỰC HIỆN

Bước 1: Khai Báo thư viện

```
import numpy as np
from tensorflow import keras
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD,Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.utils import to_categorical
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
```

Bước 2: Liên kết Google drive

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

Bước 3: Tăng Cường dữ liệu đầu vào

```
train = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
train_data = '/content/drive/MyDrive/Colab Notebooks/5 kinds of flower/training_data/flowers'
validation = ImageDataGenerator(rescale=1./255)
valid_data = '/content/drive/MyDrive/Colab Notebooks/5 kinds of flower/training_data/flowers'
```

```
train_dataset = train.flow_from_directory(train_data, target_size = (256,256), batch_size = 10, class_mode = 'categorical')
validation_dataset = validation.flow_from_directory(valid_data, target_size = (256,256), batch_size = 10, class_mode = 'categorical')
```

```
Found 4235 images belonging to 5 classes.
Found 4235 images belonging to 5 classes.
```

Bước 4: Xây dựng mô hình CNN

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same',input_shape=(256,256,3)))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(64,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())
model.add(Dense(128,activation='relu',kernel_initializer='he_uniform',))
model.add(Dense(5,activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 256, 256, 32)      896

 conv2d_5 (Conv2D)           (None, 256, 256, 32)      9248

 max_pooling2d_2 (MaxPooling  (None, 128, 128, 32)     0
 2D)

 conv2d_6 (Conv2D)           (None, 128, 128, 64)      18496

 conv2d_7 (Conv2D)           (None, 128, 128, 64)      36928

 max_pooling2d_3 (MaxPooling  (None, 64, 64, 64)       0
 2D)

 conv2d_8 (Conv2D)           (None, 64, 64, 128)       73856

 conv2d_9 (Conv2D)           (None, 64, 64, 128)       147584

 max_pooling2d_4 (MaxPooling  (None, 32, 32, 128)      0
 2D)

 flatten_1 (Flatten)         (None, 131072)            0

 dense_2 (Dense)             (None, 256)               33554688

 dense_3 (Dense)             (None, 11)                2827

=================================================================
Total params: 33,844,523
Trainable params: 33,844,523
Non-trainable params: 0
```

Bước 5: Huấn luyện mô hình

```
from tensorflow.keras.optimizers import SGD
opt = SGD(lr=0.001, momentum=0.9) # lr learning rate : tốc độ học
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_dataset,epochs=20,batch_size=32,validation_data=validation_dataset,verbose=1)
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.SGD.
Epoch 1/20
424/424 [==============================] - 2272s 5s/step - loss: 2.1999 - accuracy: 0.4642 - val_loss: 1.0605 - val_accuracy: 0.5872
Epoch 2/20
424/424 [==============================] - 106s 250ms/step - loss: 0.9867 - accuracy: 0.6224 - val_loss: 0.8080 - val_accuracy: 0.6978
Epoch 3/20
424/424 [==============================] - 105s 249ms/step - loss: 0.8797 - accuracy: 0.6720 - val_loss: 0.8070 - val_accuracy: 0.7063
Epoch 4/20
424/424 [==============================] - 105s 247ms/step - loss: 0.8151 - accuracy: 0.7011 - val_loss: 0.9426 - val_accuracy: 0.6864
Epoch 5/20
424/424 [==============================] - 105s 248ms/step - loss: 0.7600 - accuracy: 0.7259 - val_loss: 0.6447 - val_accuracy: 0.7665
Epoch 6/20
424/424 [==============================] - 124s 293ms/step - loss: 0.7143 - accuracy: 0.7450 - val_loss: 0.6057 - val_accuracy: 0.7792
Epoch 7/20
424/424 [==============================] - 104s 246ms/step - loss: 0.6821 - accuracy: 0.7577 - val_loss: 0.5855 - val_accuracy: 0.7894
Epoch 8/20
424/424 [==============================] - 123s 291ms/step - loss: 0.6604 - accuracy: 0.7747 - val_loss: 0.5777 - val_accuracy: 0.7934
Epoch 9/20
424/424 [==============================] - 124s 292ms/step - loss: 0.6318 - accuracy: 0.7769 - val_loss: 0.4539 - val_accuracy: 0.8425
Epoch 10/20
424/424 [==============================] - 104s 245ms/step - loss: 0.6027 - accuracy: 0.7889 - val_loss: 0.5004 - val_accuracy: 0.8286
Epoch 11/20
424/424 [==============================] - 104s 246ms/step - loss: 0.5708 - accuracy: 0.7981 - val_loss: 0.4766 - val_accuracy: 0.8340
Epoch 12/20
424/424 [==============================] - 103s 244ms/step - loss: 0.5551 - accuracy: 0.8045 - val_loss: 0.4467 - val_accuracy: 0.8394
Epoch 13/20
424/424 [==============================] - 105s 247ms/step - loss: 0.5350 - accuracy: 0.8182 - val_loss: 0.4139 - val_accuracy: 0.8654
Epoch 14/20
424/424 [==============================] - 123s 290ms/step - loss: 0.5425 - accuracy: 0.8246 - val_loss: 0.3454 - val_accuracy: 0.8751
Epoch 15/20
424/424 [==============================] - 104s 246ms/step - loss: 0.5298 - accuracy: 0.8255 - val_loss: 0.4865 - val_accuracy: 0.8491
Epoch 16/20
424/424 [==============================] - 103s 243ms/step - loss: 0.5141 - accuracy: 0.8300 - val_loss: 0.3507 - val_accuracy: 0.8725
Epoch 17/20
424/424 [==============================] - 104s 245ms/step - loss: 0.5176 - accuracy: 0.8267 - val_loss: 0.3191 - val_accuracy: 0.8947
Epoch 18/20
424/424 [==============================] - 103s 243ms/step - loss: 0.5131 - accuracy: 0.8326 - val_loss: 0.4390 - val_accuracy: 0.8569
Epoch 19/20
424/424 [==============================] - 104s 246ms/step - loss: 0.5176 - accuracy: 0.8401 - val_loss: 0.3328 - val_accuracy: 0.8876
Epoch 20/20
424/424 [==============================] - 125s 294ms/step - loss: 0.5073 - accuracy: 0.8411 - val_loss: 0.3085 - val_accuracy: 0.8914
```

Bước 6 :Đánh giá độ chính xác

```
flower5_model = load_model('flower5.h5')
```

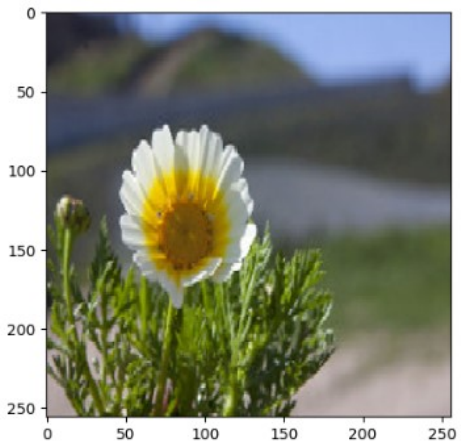
```
score=flower5_model.evaluate(validation_dataset,verbose=1)
print('Test loss = ',score[0])
print('Test accuracy = ',score[1])
```

```
424/424 [==============================] - 23s 55ms/step - loss: 0.3085 - accuracy: 0.8914
Test loss =  0.30849409103393555
Test accuracy =  0.8913813233375549
```

4.Kết quả

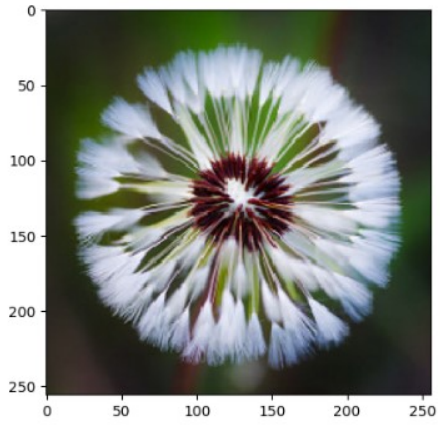
```
img = load_img('/content/drive/MyDrive/Colab Notebooks/5 kinds of flower/testing_data/daisy/daisy.jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['daisy','rose','sunflower','tulip','dandelion']
print('Predict is: ',Label[int(np.argmax(flower5_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 370ms/step
Predict is:  daisy
```



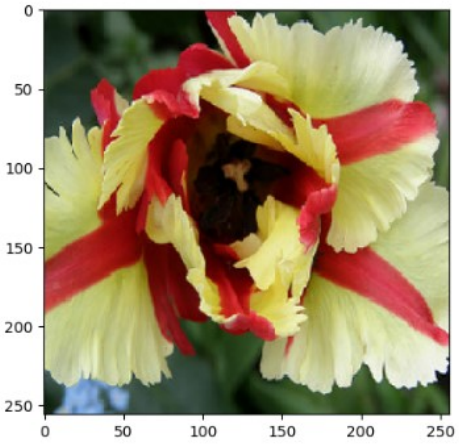

```
img = load_img('/content/drive/MyDrive/Colab Notebooks/5 kinds of flower/testing_data/dandelion/dandelion.jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['daisy','rose','sunflower','tulip','dandelion']
print('Predict is: ',Label[int(np.argmax(flower5_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 29ms/step
Predict is:  rose
```



```
img = load_img('/content/drive/MyDrive/Colab Notebooks/5 kinds of flower/testing_data/tulip/tulip.jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['daisy','rose','sunflower','tulip','dandelion']
print('Predict is: ',Label[int(np.argmax(flower5_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 58ms/step
Predict is:  dandelion
```



5.Kết luận

Với bài tập nhận diện 5 loại hoa này ,thuật toán CNN cho ra tỉ tệ chính xác khoảng 89% với 20 lần học .Cho thấy được tốc độ xữ lý của thuật toán CNN Nhanh và chính xác

BÀI TẬP SỐ 3 : NHẬN DIỆN THÀNH VIÊN TRONG LỚP TỪ HÌNH ẢNH KHUÔN MẶT

1.GIỚI THIỆU

Khuôn mặt là phần quan trọng nhất của cơ thể con người, nó giúp ta giao tiếp, thể hiện cảm xúc và cá tính của mình. Khuôn mặt bao gồm nhiều phần như trán, mắt,mũi,miệng,cằmvà tai.

Mỗi người có một khuôn mặt độc đáo, khác nhau về hình dạng, màu sắc, kích thước và đặc điểm. Nhưng ngoài những yếu tố đó, khuôn mặt còn mang đến nhiều thông tin về sức khỏe và tuổi tác của một người.

Ví dụ, nếp nhăn, đường nét trên khuôn mặt có thể thể hiện tuổi tác của một người. Tình trạng da và màu sắc của khuôn mặt có thể phản ánh sức khỏe và lối sống của một người.

Ngoài ra, khuôn mặt còn được coi là "gương mặt" của một cá nhân, nó cho thấy tính cách, tâm trạng và cảm xúc của một người. Một nụ cười tươi cũng có thể làm cho khuôn mặt trở nên đẹp hơn và gần gũi hơn với người khác.

2.PHƯƠNG PHÁP LUẬN

2.1 Thuật toán CNN

Thuật toán CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo được sử dụng trong bài toán phân loại ảnh và xử lý ảnh. Thuật toán này giúp cho việc nhận dạng, phân loại và xử lý ảnh trở nên chính xác hơn.

CNN bao gồm nhiều lớp, mỗi lớp đóng vai trò trong quá trình xử lý và trích xuất đặc trưng của ảnh. Bên trong mỗi lớp là một tập hợp các bộ lọc (filters), được áp dụng trên ảnh đầu vào để tạo ra các đặc trưng cụ thể của ảnh.

Các lớp trong CNN bao gồm:

Lớp Input: Lớp này chứa ảnh đầu vào và có kích thước tùy ý.

Lớp Convolution: Lớp này sử dụng bộ lọc để trích xuất đặc trưng của ảnh. Bộ lọc được áp dụng lên ảnh đầu vào và tạo ra các ma trận tích chập (convolution matrix). Các ma trận tích chập này có thể hiểu như là các bản đồ đặc trưng (feature maps) của ảnh đầu vào.

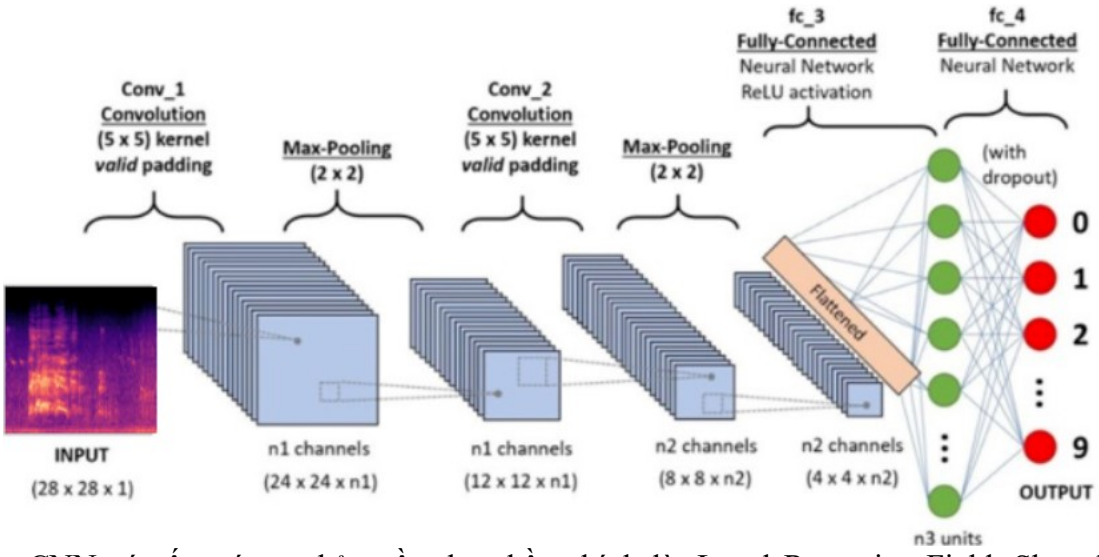
Lớp Activation: Lớp này áp dụng hàm kích hoạt (activation function) lên các bản đồ đặc trưng để tạo ra bản đồ đặc trưng kích hoạt (activated feature maps). Hàm kích hoạt thường được sử dụng là hàm ReLU (Rectified Linear Unit).

Lớp Pooling: Lớp này giảm kích thước của bản đồ đặc trưng bằng cách chọn ra các giá trị quan trọng nhất. Các phép toán thường được sử dụng trong lớp này là Max Pooling hoặc Average Pooling.

Lớp Fully Connected: Lớp này được sử dụng để kết nối các bản đồ đặc trưng đã được trích xuất từ các lớp trước đó và phân loại ảnh. Lớp này sử dụng một hoặc nhiều lớp fully connected để tính toán đầu ra cuối cùng.

CNN được sử dụng rộng rãi trong các bài toán liên quan đến xử lý ảnh và thị giác máy tính, chẳng hạn như nhận dạng khuôn mặt, phân loại chủ đề của ảnh, và phát hiện đối tượng trong ảnh.

2.2 Cấu trúc mạng CNN



CNN có cấu trúc cơ bản gồm ba phần chính là: Local Receptive Field, Shared Weights And Bias và Pooling.

Local Receptive Field (trường tiếp nhận cục bộ)

Đây được xem là lớp giúp bạn có thể tách lọc các dữ liệu, thông tin của ảnh và chọn được những vùng ảnh có giá trị sử dụng nhất.

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

Tuy nhiên trong CNN chúng ta không làm như vậy mà chúng ta chỉ kết nối trong một vùng nhỏ của các neuron đầu vào như một filter có kích thước 5×5 tương ứng (28- 5 + 1) 24 điểm ảnh đầu vào. Mỗi một kết nối sẽ học một trọng số và mỗi neuron ẩn sẽ học một bias. Mỗi một vùng 5×5 đấy gọi là một trường tiếp nhận cục bộ

Shared Weights And Bias (trọng số chia sẻ)

Chức năng chính của lớp này là hỗ trợ bạn làm giảm tối đa số lượng những tham số trong mạng CNN. Vì trong mỗi Convolution sẽ bao gồm các Feature Map khác nhau, mỗi Feature Map lại giúp Detect một vài Feature trong ảnh.

Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-rontrong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Tóm lại, một convolutional layer bao gồm các feature map khác nhau. Mỗi một feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN

Pooling Layer (lớp tổng hợp)

Đây được xem gần như là lớp cuối cùng trước khi đưa ra kết quả trong CNN. Chính vì thế, để có được kết quả dễ hiểu và dễ sử dụng nhất thì Pooling Layer có nhiệm vụ làm đơn giản hóa các thông tin đầu ra. Nghĩa là, sau khi hoàn thành quá trình tính toán và quét các lớp thì sẽ đi đến Pooling Layer nhằm lượt bớt các không tin không cần thiết và cho ra kết quả mà chúng ta đang cần.

3.THỰC HIỆN

Bước 1: Khai Báo thư viện

```
import numpy as np
from tensorflow import keras
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD,Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.utils import to_categorical
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
```

Bước 2: Liên kết Google drive

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

Bước 3: Tăng cường dữ liệu đầu vào

```
train = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
train_data = '/content/drive/MyDrive/Colab Notebooks/Image'
validation = ImageDataGenerator(rescale=1./255)
valid_data = '/content/drive/MyDrive/Colab Notebooks/Image'
```

```
train_dataset = train.flow_from_directory(train_data, target_size = (256,256), batch_size = 10, class_mode = 'categorical')
validation_dataset = validation.flow_from_directory(valid_data, target_size = (256,256), batch_size = 10, class_mode = 'categorical')
```

```
Found 455 images belonging to 10 classes.
Found 455 images belonging to 10 classes.
```

Bước 4: Xây dựng mô hình CNN

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same',input_shape=(256,256,3)))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(32,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(64,(3,3),activation = 'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())
model.add(Dense(128,activation='relu',kernel_initializer='he_uniform',))
model.add(Dense(10,activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 256, 256, 32)      896

 conv2d_7 (Conv2D)           (None, 256, 256, 32)      9248

 max_pooling2d_3 (MaxPooling  (None, 128, 128, 32)     0
 2D)

 conv2d_8 (Conv2D)           (None, 128, 128, 32)      9248

 conv2d_9 (Conv2D)           (None, 128, 128, 32)      9248

 max_pooling2d_4 (MaxPooling  (None, 64, 64, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 64, 64, 64)        18496

 conv2d_11 (Conv2D)          (None, 64, 64, 64)        36928

 max_pooling2d_5 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 65536)             0

 dense_2 (Dense)             (None, 128)               8388736

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 8,474,090
Trainable params: 8,474,090
Non-trainable params: 0
_________________________________________________________________
```

Bước 5: Huấn luyện mô hình

```
from tensorflow.keras.optimizers import SGD
opt = SGD(lr=0.001, momentum=0.9) # lr learning rate : tốc độ học
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.SGD.
```

```
history = model.fit(train_dataset,epochs=10,batch_size=20,validation_data=validation_dataset,verbose=1)
```

```
Epoch 1/10
46/46 [==============================] - 229s 5s/step - loss: 4.8297 - accuracy: 0.3319 - val_loss: 1.3887 - val_accuracy: 0.5121
Epoch 2/10
46/46 [==============================] - 259s 6s/step - loss: 1.0183 - accuracy: 0.6593 - val_loss: 0.2301 - val_accuracy: 0.9319
Epoch 3/10
46/46 [==============================] - 265s 6s/step - loss: 0.7486 - accuracy: 0.8308 - val_loss: 0.0812 - val_accuracy: 0.9736
Epoch 4/10
46/46 [==============================] - 224s 5s/step - loss: 0.5755 - accuracy: 0.8747 - val_loss: 0.2848 - val_accuracy: 0.9143
Epoch 5/10
46/46 [==============================] - 259s 6s/step - loss: 0.3807 - accuracy: 0.9297 - val_loss: 0.0521 - val_accuracy: 0.9934
Epoch 6/10
46/46 [==============================] - 257s 6s/step - loss: 1.3789 - accuracy: 0.9407 - val_loss: 0.0146 - val_accuracy: 0.9956
Epoch 7/10
46/46 [==============================] - 257s 6s/step - loss: 0.5978 - accuracy: 0.9385 - val_loss: 0.2743 - val_accuracy: 0.9253
Epoch 8/10
46/46 [==============================] - 259s 6s/step - loss: 0.4177 - accuracy: 0.9495 - val_loss: 0.2872 - val_accuracy: 0.9319
Epoch 9/10
46/46 [==============================] - 260s 6s/step - loss: 0.1098 - accuracy: 0.9824 - val_loss: 0.2945 - val_accuracy: 0.9473
Epoch 10/10
46/46 [==============================] - 219s 5s/step - loss: 0.3060 - accuracy: 0.9560 - val_loss: 0.0168 - val_accuracy: 0.9956
```

Buoc 6:Đánh Giá Độ Chính Xác

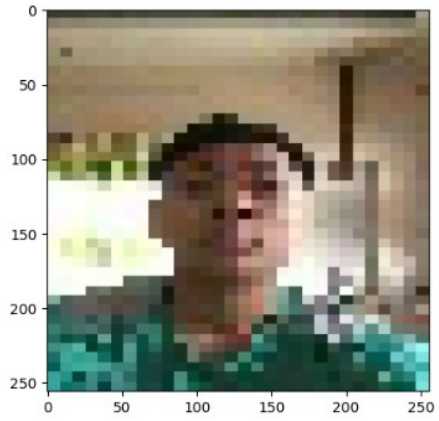
```
score=Face10_model.evaluate(validation_dataset,verbose=1)
print('Test loss = ',score[0])
print('Test accuracy = ',score[1])
```

```
46/46 [==============================] - 46s 981ms/step - loss: 0.0168 - accuracy: 0.9956
Test loss =  0.016766432672739 03
Test accuracy =  0.995604395866394
```

4.KẾT QUẢ

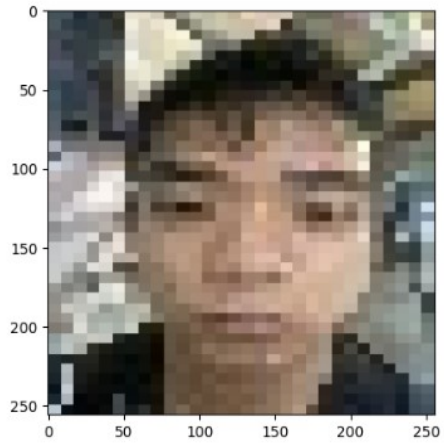
```
img = load_img('/content/drive/MyDrive/Colab Notebooks/Face images /Testing_data/Hong Nhan/nhn11.jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['Dat','Tan','Tan Thanh','Ba Trong','Minh Thanh','Nhut','Nha','Tuan Giang','Nhan','phat']
print('Predict is: ',Label[int(np.argmax(Face10_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 273ms/step
Predict is:  Nhan
```



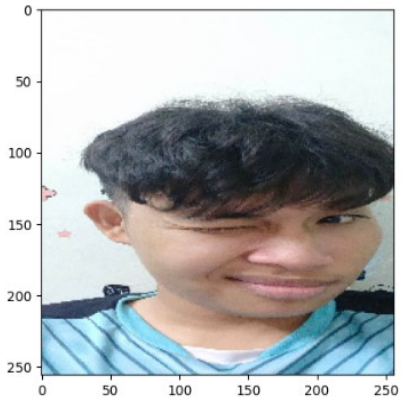
```
img = load_img('/content/drive/MyDrive/Colab Notebooks/Face images /Testing_data/Minh Thanh/Thanh1.jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['Dat','Tan','Tan Thanh','Ba Trong','Minh Thanh','Nhut','Nha','Tuan Giang','Nhan','phat']
print('Predict is: ',Label[int(np.argmax(Face10_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 151ms/step
Predict is:  Tan Thanh
```




```
img = load_img('/content/drive/MyDrive/Colab Notebooks/Image/Dat/330271565_1461410234596877_2261443984632201285_n.jpg',target_size=(256,256))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,256,256,3)
img = img.astype('float32')
img = img/255
Label = ['Tan','Dat','Tan Thanh','Ba Trong','Minh Thanh','Nhut','Nha','Tuan Giang','Nhan','phat']
print('Predict is: ',Label[int(np.argmax(Face10_model.predict(img),axis=-1))])
```

```
1/1 [==============================] - 0s 157ms/step
Predict is:  Dat
```



5.KẾT LUẬN

Nhận diện khuôn mặt thông qua thuật toán CNN cho thấy độ chính Xác cao 99%.Mô hình đáp ứng được mục tiêu và cần bổ sung thêm nhiều dữ liệu để thuật toán chính xác nhất và đạt hiệu quả tốt nhất