ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Bài tập 01

MÔN: NHẬN DẠNG

Tp Hồ Chí Minh 2022

ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

NGUYỄN THANH TÙNG

BÀI TẬP 02: Nhận dạng mặt người

Môn: Nhận dạng

MSSV: 20120617

Tên: Nguyễn Thanh Tùng

Email: tunghcltt@gmail.com

GIÁO VIÊN

Thầy Lê Hoàng Thái

Tp Hồ Chí Minh 2022

Mục lục

I. Phân tích

1) Phân tích bài toán

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

2) CNN:

a. Convolution là gì

- Convolutional (convolution) là một phép toán trong xử lý ảnh và thị giác máy tính. Nó được sử dụng trong mạng neural convolutional (CNN) để xác định các đặc trưng trong ảnh.
- Trong quá trình convolution, một kernel (hay filter) nhỏ được trượt qua các vùng con của ảnh đầu vào để tạo ra một feature map (bản đồ đặc trưng). Kernel là một ma trận nhỏ chứa các trọng số. Quá trình convolution tính toán tích chập giữa kernel và vùng con của ảnh để tạo ra giá trị mới cho feature map.
- Quá trình convolution giúp tìm ra các đặc trưng cục bộ trong ảnh, như cạnh, góc, hoặc các đặc trưng cao cấp hơn như khuôn mặt. Bằng cách áp dụng nhiều kernel khác nhau và sử dụng các lớp convolutional liên tiếp, CNN có khả năng tự động học các đặc trưng phức tạp từ dữ liệu ảnh.

b. Cấu trúc mạng CNN

- Mạng CNN 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. 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.

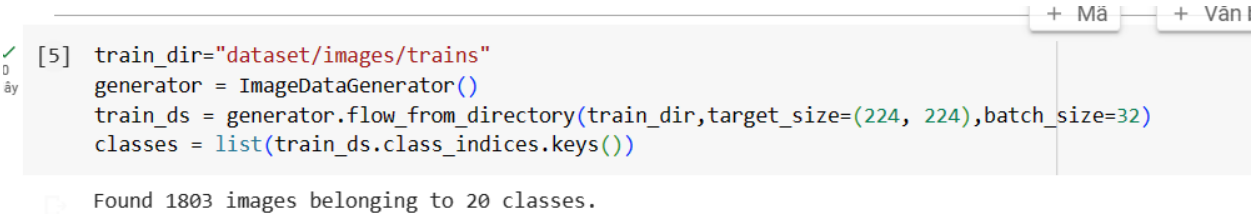
II. Cài đặt

1. Xử lý dữ liệu

- Dataset bao gồm thư mục images. Thư mục images chứa 2 thư mục là trains và tests. 2 thư mục này chứa các hình ảnh dùng để huấn luyện và kiểm tra hoạt d0on65g của mô hình

- Đưa file dataset.zip lên google drive sau đó giải nén và sử dụng trong môi trường google colab

- Tạo dữ liệu huấn luyện từ các ảnh trong thư mục trains



2. Cài đặt mô hình

```
model = Sequential()
model.add(Conv2D(32, kernel_size = (3, 3), activation='relu', input_shape=(224,224,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(96, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
#model.add(Dropout(0.3))
model.add(Dense(len(classes),activation='softmax'))
```

Các lớp trong mô hình:

- Lớp Conv2D: Lớp Conv2D áp dụng các bộ lọc có kích thước (kernel_size) là 3x3 lên đầu vào 2D (ảnh) để tìm các đặc trưng. Có 32 bộ lọc (filters) được sử dụng và hàm kích hoạt (activation function) là hàm ReLU (Rectified Linear Unit).Đầu vào của lớp Conv2D có kích thước (input_shape) là (224, 224, 3), tương ứng với kích thước ảnh đầu vào là 224x224 pixel và 3 kênh màu (RGB).
- Lớp MaxPooling2D: Mục tiêu của lớp này là giảm kích thước không gian và trích xuất các đặc trưng quan trọng.
- Lớp BatchNormalization: được sử dụng để chuẩn hóa đầu vào của lớp trước đó, giúp tăng tốc độ học và giảm hiện tượng quá khớp.
- Lớp Flatten: được sử dụng để chuyển đổi đầu ra từ các lớp trước đó thành một vectơ 1D để có thể kết nối với các lớp fully connected.

- Lớp Dense: là một fully connected layer với 128 đơn vị đầu ra. Hàm kích hoạt của lớp là hàm ReLU (Rectified Linear Unit), giúp tạo ra một không gian phi tuyến tính và tăng tính phi tuyến của mô hình.

3. Training

```
history = model.fit(train_ds,epochs= 30, batch_size=32)
```

- train_ds: Dữ liệu huấn luyện

- epochs: Số lượng vòng lặp được sử dụng trong quá trình huấn luyện. Mỗi epoch tương ứng với việc đi qua toàn bộ dữ liệu huấn luyện một lần.

- batch_size: Kích thước của các batch dữ liệu được sử dụng trong quá trình huấn luyện.

Kết quả của quá trình huấn luyện được trả về dưới dạng một đối tượng history. Đối tượng này chứa thông tin về các độ đo và hàm mất mát trong quá trình huấn luyện, cho phép kiểm tra và phân tích hiệu suất của mô hình sau khi huấn luyện hoàn thành.

```
    history = model.fit(train_ds,epochs= 30, batch_size=32)
```

```
Epoch 1/30
57/57 [==============================] - 197s 3s/step - loss: 2.2624 - accuracy: 0.3611
Epoch 2/30
57/57 [==============================] - 194s 3s/step - loss: 1.1582 - accuracy: 0.6500
Epoch 3/30
57/57 [==============================] - 194s 3s/step - loss: 0.5757 - accuracy: 0.8297
Epoch 4/30
57/57 [==============================] - 191s 3s/step - loss: 0.3156 - accuracy: 0.9174
Epoch 5/30
57/57 [==============================] - 197s 3s/step - loss: 0.1752 - accuracy: 0.9490
Epoch 6/30
57/57 [==============================] - 191s 3s/step - loss: 0.0971 - accuracy: 0.9806
Epoch 7/30
57/57 [==============================] - 191s 3s/step - loss: 0.0586 - accuracy: 0.9900
Epoch 8/30
57/57 [==============================] - 191s 3s/step - loss: 0.0522 - accuracy: 0.9884
Epoch 9/30
57/57 [==============================] - 200s 4s/step - loss: 0.0229 - accuracy: 0.9989
Epoch 10/30
57/57 [==============================] - 190s 3s/step - loss: 0.0115 - accuracy: 0.9994
Epoch 11/30
57/57 [==============================] - 201s 4s/step - loss: 0.0087 - accuracy: 0.9994
Epoch 12/30
57/57 [==============================] - 199s 3s/step - loss: 0.0067 - accuracy: 1.0000
Epoch 13/30
...
```

4. Triển khai mô hình

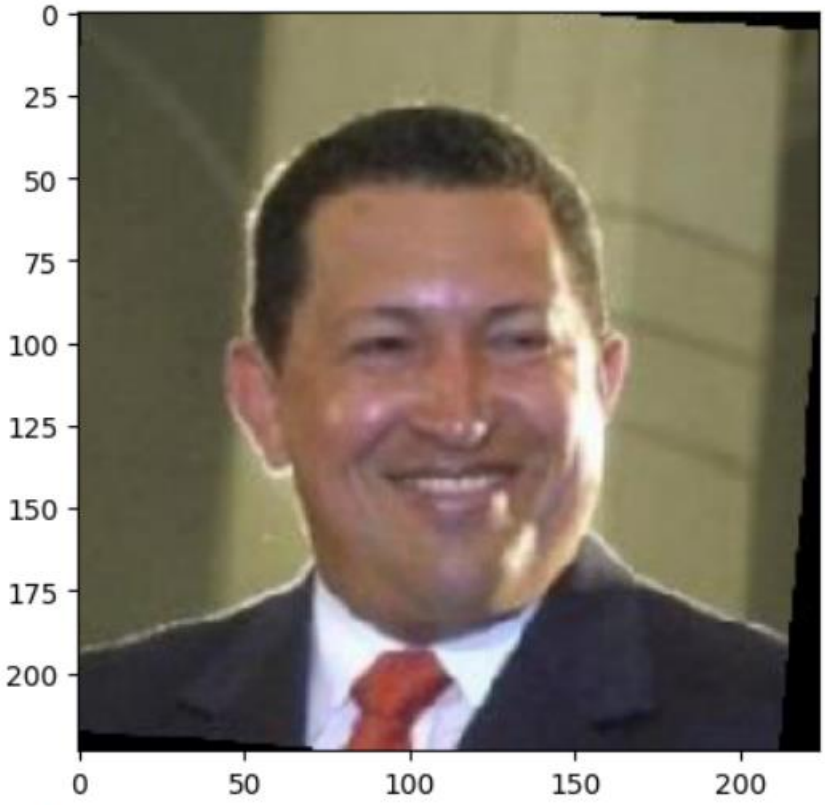
```
def predict_image(image_path):
    img = image.load_img(image_path, target_size=(224,224,3))
    plt.imshow(img)
    plt.show()
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])
    pred = model.predict(images, batch_size=32)
    print("Actual: "+(image_path.split("/")[-1]).split("_")[0]+"_"+(image_path.split("/")[-1]).split("_")[1])
    print("Predicted: "+classes[np.argmax(pred)])
```

```
predict_image("dataset/images/trains/Hugo_Chavez/Hugo_Chavez_0019.jpg")
```

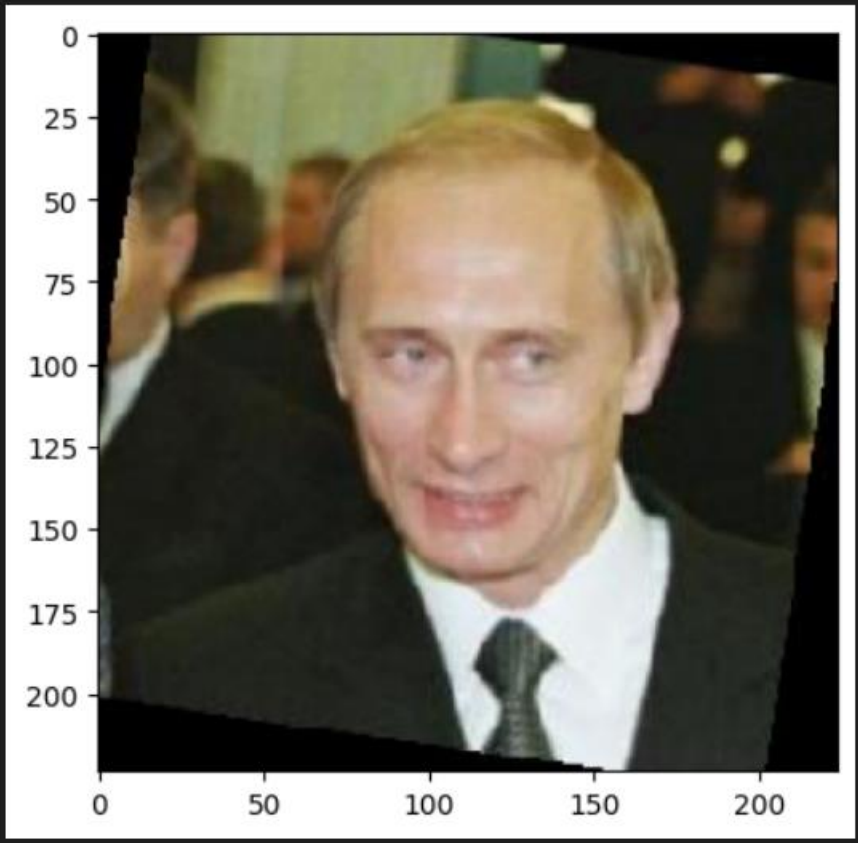
Đưa hình ảnh từ file tests để mô hình nhận diện.

Dự đoán lớp của hình ảnh đó bằng mô hình CNN và in ra lớp thực tế và lớp dự đoán.

1 số kết quả:




```
1/1 [==============================] - 0s 79ms/step
Actual: HugoChavez
Predicted: Hugo_Chavez
```




```
1/1 [==============================] - 0s 42ms/step
Actual: Vladimir_Putin
Predicted: Vladimir_Putin
```