ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ:
PHÂN LOẠI HÌNH ẢNH CÓ TRONG THƯ VIỆN ĐIỆN THOẠI

SINH VIÊN THỰC HIỆN: ĐỖ TRÍ GIA BẢO - 21520603

TRƯƠNG LÊ DIỄN – 21520189

NGUYỄN THÀNH VINH – 21522793

LỚP: CS114.O11

GIẢNG VIÊN HƯỚNG DẪN:

PhGS.TS Lê Đình Duy

ThS Phạm Nguyễn Trường An

TP. HỒ CHÍ MINH – Tháng 1 năm 2024

Tóm tắt đề tài:

Dự án Machine Learning mà nhóm thực hiện tập trung vào việc áp dụng các phương pháp học máy, đặc biệt là mô hình supervised learning, vào bài toán phân loại hình ảnh. Mục tiêu cốt lõi của dự án là xây dựng một hệ thống tự động có khả năng nhận diện và phân loại các hình ảnh, sau đó tự động sắp xếp chúng vào các thư mục hoặc danh mục có nhãn tương ứng.

Quá trình này bao gồm các bước chính như sau: thu thập dữ liệu hình ảnh, tiền xử lý dữ liệu để chuẩn bị cho việc huấn luyện mô hình, xây dựng và huấn luyện một mạng nơ-ron tích chập (CNN) bằng cách sử dụng framework Keras trong TensorFlow, và cuối cùng là đánh giá và điều chỉnh mô hình để tối ưu hóa hiệu suất phân loại.

Đề tài cung cấp một giải pháp tự động hoàn toàn cho việc phân loại hình ảnh, giúp tiết kiệm thời gian và công sức so với việc thủ công. Ngoài ra, nó cũng có thể được mở rộng để áp dụng vào nhiều lĩnh vực khác nhau như phát hiện vật thể, nhận dạng khuôn mặt, hoặc phân loại đối tượng trong các bức ảnh y tế, công nghiệp, hoặc môi trường. Điều này mở ra nhiều tiềm năng ứng dụng trong thực tế và nghiên cứu trong tương lai.

Link dẫn đến github repository của nhóm:
https://github.com/VinhZa/CS114.O11_21522793.git

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

0. UPDATE SAU VẤN ĐÁP

Nhóm không có cập nhật gì sau vấn đáp.

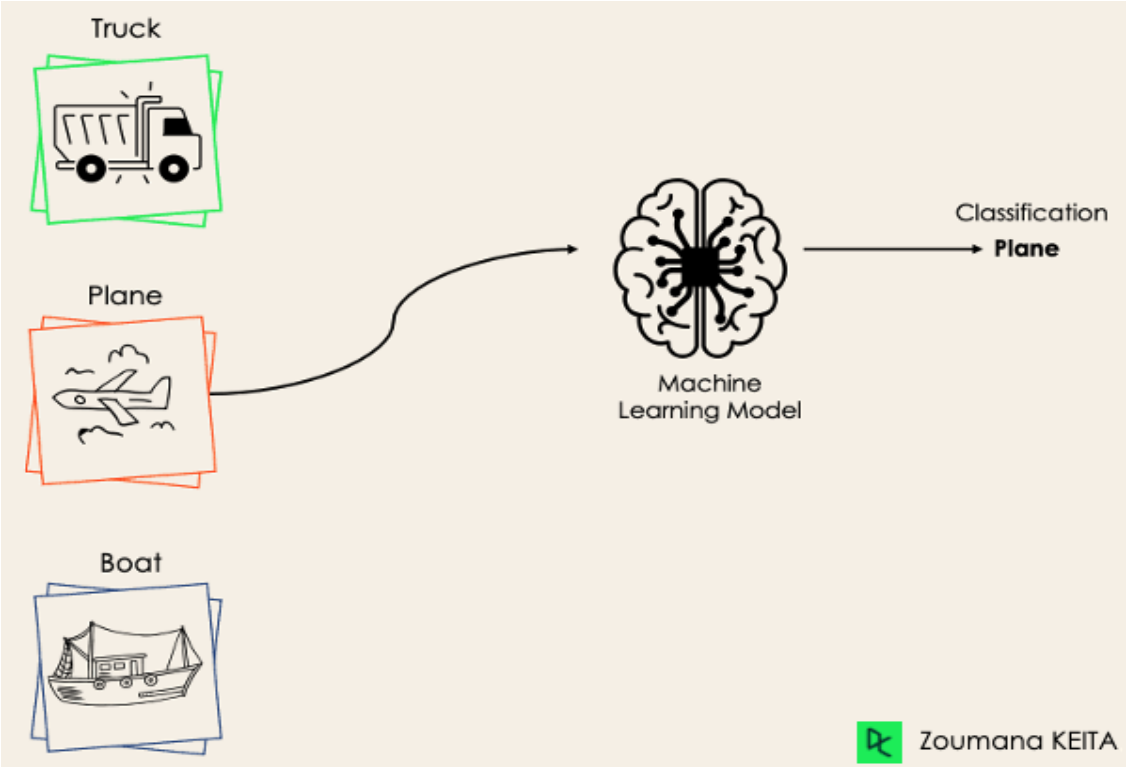
1. TỔNG QUAN ĐỒ ÁN

1.1. Mô tả đề tài và ngữ cảnh ứng dụng

Với bối cảnh ứng dụng là trên một thư viện chứa ảnh của điện thoại, khi người sử dụng ngẫu nhiên mở điện thoại lên và chụp hoặc lưu một ảnh thì ảnh sẽ được đẩy vào thư vi

ện chính. Nhưng sau một thời gian khi số lượng ảnh nạp vào quá lớn và bạn cần lấy ra ảnh cần thiết là rất khó khăn và mất thời gian vì có quá nhiều nội dung. Từ đó, nhóm có ý tưởng sẽ hiện thực một mô hình như sau:

- Input: một ảnh có nội dung ngẫu nhiên
- Output: label của ảnh, sau đó dựa trên label đó sẽ đưa vào folder có label tương ứng nhằm phân nhóm các ảnh.



Hình 1 – Mô tả khái quát về đề tài

Đề tài đồ án của nhóm với mục tiêu là hiện thực một hệ thống có

chức năng cụ thể như sau: Khi người dùng đưa ảnh vào với các nội dung trong ảnh là ngẫu nhiên, mô hình máy học sẽ tự động nhận dạng và hình ảnh sẽ được đưa vào thư mục có label tương ứng với ảnh.

Việc hiện thực một mô hình như trên giúp cho việc quản lí các ảnh trong thư viện điện thoại trở nên dễ dàng hơn thông qua cách tìm kiếm các ảnh nhanh chóng hơn, hạn chế việc lặp lại của các ảnh,…

1.2. Mô tả bộ dữ liệu

Số lượng và chủ đề: Tập dữ liệu sử dụng để huấn luyện mô hình sẽ bao gồm 4 label thông dụng như là xe hơi, hoa cỏ, ảnh khuôn mặt(selfie), thú nuôi. Mỗi nội dung tương tự sẽ có khoảng 500 ảnh được chia thành 2 tập train và validation với tỉ lệ là 4:1.

Yêu cầu: Với tính chất là bài toán của mô hình supervised learning, vì vậy các dữ liệu nạp vào cho quá trình huấn luyện là những input có nhãn. Các folder riêng biệt có tên tương ứng với các label cần dự đoán cho mô hình. Trong quá trình lấy dữ liệu để training thì label sẽ được lấy dựa trên tên của mỗi folder.

Thu thập dữ liệu: Với độ rộng của các chủ đề sẽ xuất hiện trong thư viện điện thoại, nhóm sẽ đưa ra những chủ đề cụ thể và dễ xuất hiện **đối với nhóm**, từ đó hiện thực thành dữ liệu training cho mô hình. Các nhóm cụ thể là xe hơi, hoa lá, pet, ảnh có mặt người(ảnh selfie). Với các chủ đề trên, các ảnh được nhóm thu thập từ các nguồn như các trang mạng xã hội, các trang web, ảnh chụp từ camera,…

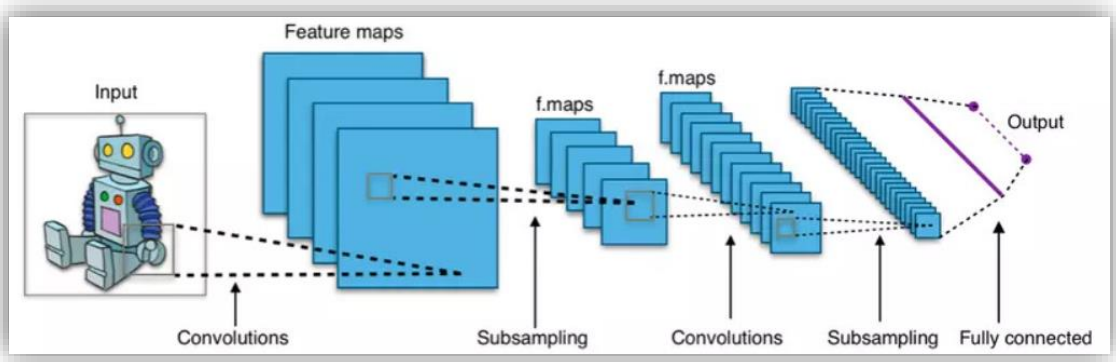
2. HIỆN THỰC ĐỒ ÁN

2.1. Khái quát về bài toán Image Classification

- Bài toán này là một quá trình trong lĩnh vực thị giác máy tính, nơi mà mục tiêu là phân loại **<u>một hình ảnh</u>** vào một số lớp xác định trước
- Thu thập dữ liệu: Xây dựng một bộ dữ liệu đa dạng chứa hình ảnh thuộc các lớp cần phân loại
- Tiền xử lý dữ liệu: Resize hình ảnh về kích thước thích hợp để đầu vào cho mô hình. Chuẩn hóa dữ liệu để đảm bảo mô hình đào tạo và kiểm thử hoạt động hiệu quả

- Sử dụng các mô hình sâu(deep learning) như Convolutional Neural Networks(CNNs) được thiết kế đặc biệt cho việc xử lý hình ảnh. Chọn kiến trúc mô hình phù hợp với vấn đề cụ thể và kích thước dữ liệu
- Đào tạo mô hình: Sử dụng bộ dữ liệu đã chuẩn bị để đào tạo mô hình
- Kiểm thử và đánh giá: Đánh giá hiệu suất của mô hình. Đo lường các chỉ số như độ chính xác(accuracy), độ nhạy(recall),..

2.2. Khái quát về CNN (Convolutional Neural Network)



Hình 2 – Mô tả khái quát mô hình CNN

Convolutional Neural Networks (CNNs) được ứng dụng rộng rãi trong việc phân loại ảnh trong máy học vì các lý do sau:

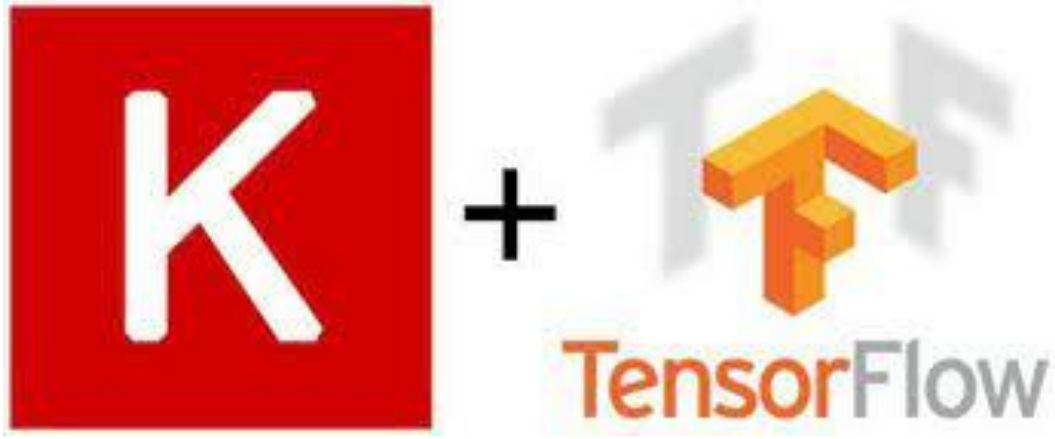
- Convolutional Neural Network hay còn được gọi là CNNS mạng nơ-ron tích chập, là một trong những mô hình Deep Learning cực kỳ tiên tiến, bởi chúng cho phép bạn xây dựng những hệ thống có độ chính xác cao và thông minh
- Cấu trúc tổ chức của ảnh: CNN được thiết kế để hiệu quả trong việc xử lý dữ liệu ảnh bằng cách sử dụng các lớp convolutional để trích xuất đặc trưng từ các vùng nhỏ của ảnh. Điều này phù hợp với cấu trúc tổ chức của dữ liệu hình ảnh, trong đó các đặc trưng quan trọng thường được tìm thấy ở các vùng cục bộ của hình ảnh.
- Chia sẻ trọng số và tỷ lệ tham số: CNN sử dụng các lớp convolutional và pooling để chia sẻ trọng số, giúp giảm số lượng tham số cần học và tăng tốc quá trình huấn luyện. Điều này rất quan trọng khi làm việc với dữ liệu ảnh, với kích thước lớn và số lượng tham số cần thiết có thể rất lớn nếu mỗi trọng số được kết nối một cách đầy đủ.
- Khả năng học đặc trưng cấp cao: CNN có khả năng học các đặc trưng cấp cao từ dữ liệu hình ảnh thông qua việc tổng hợp các đặc trưng cơ

bản, như các cạnh, góc, hoặc mảng màu, từ các lớp convolutional sâu. Điều này cho phép chúng hiệu quả trong việc nhận diện các đặc trưng phức tạp, như hình dạng và cấu trúc tổ chức, trong dữ liệu ảnh.

- Khả năng tự động hóa quy trình trích xuất đặc trưng: Thay vì cần phải thiết kế và chọn lọc các đặc trưng thủ công từ dữ liệu ảnh, CNN có thể tự động học và trích xuất các đặc trưng phù hợp cho nhiệm vụ phân loại ảnh, giảm thiểu sự can thiệp của con người và tăng cường khả năng tự động hóa của hệ thống. Những lợi ích này khiến cho CNN trở thành lựa chọn phổ biến và hiệu quả trong việc xử lý và phân loại dữ liệu ảnh trong máy học và thị giác máy tính

➔ Từ đó, nhóm chọn ứng dụng CNN cho mô hình.

2.3. Tìm hiểu về Tensorflow(Keras)



TensorFlow, một thư viện mã nguồn mở phổ biến trong lĩnh vực máy học và trí tuệ nhân tạo, cung cấp một giao diện để xây dựng, đào tạo, và triển khai mô hình máy học, bao gồm cả CNN

Trong TensorFlow, Keras là một API cao cấp và dễ sử dụng cho việc xây dựng và huấn luyện mô hình máy học, đặc biệt là mạng nơ-ron. Keras cung cấp một cách tiếp cận linh hoạt và trực quan cho việc xây dựng mô hình, giúp người dùng tập trung vào việc thiết kế mô hình mà không cần quá nhiều kiến thức về lập trình hoặc chi tiết cài đặt.

Các đặc điểm tiêu biểu của Keras phải kể đến đó là dễ sử dụng vì được thiết kế đơn giản , cấu trúc mã nguồn rõ ràng và các API trực quan; hỗ trợ nhiều backend; hỗ trợ đa dạng loại mô hình;…

➔ Do vậy, Keras được nhóm chọn làm thư viện chính, giúp đơn giản hóa quá trình xây dựng mô hình học sâu.

2.4. Hiện thực chương trình

2.4.1. Khai báo các môi trường thực thi và các thư viện cần thiết cho mô hình

```
import numpy as np
import random
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense,
BatchNormalization, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import Adam

%matplotlib inline
```

Hình 3 – Mô trường thực thi và thư viện

- import matplotlib.pyplot as plt : Thư viện này dùng để vẽ đồ thị
- from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping : thư viện này dùng để gọi hàm call back trong quá trình training
- from tensorflow.keras.preprocessing.image import ImageDataGenerator: thư viện này để gọi hàm tăng cường dữ liệu, nhằm mục đích tạo tập dữ liệu cho quá trình training
- from tensorflow.keras.preprocessing import image: dùng cho việc load image
- from tensorflow.keras.models import Sequential: xây dựng mô hình tuần tự trong Keras
- from tensorflow.keras.layers import GlobalAveragePooling2D, Dense,
- BatchNormalization, Dropout, Flatten, Conv2D, MaxPooling2D:nhập các lớp cần thiết cho CNN
- from tensorflow.keras.optimizers import Adam: import trình tối ưu hóa
- %matplotlib inline: hiển thị đồ thị trực tiếp trong notebook

2.4.2. Tinh chỉnh dữ liệu

```
im_shape = (250,250)
seed = 10
BATCH_SIZE = 16
```

Bảng 1. Định dạng hình ảnh, nạp dữ liệu

Im_shape : nhằm chuẩn hóa ảnh nạp vào với kích thước là 250x250
Kế tiếp là seed = 10, có chức năng xác định số trạng thái ngẫu nhiên sẽ được giữ nguyên qua các lần epoch, bước này giúp dữ liệu training tốt hơn và giúp hiệu suất dự đoán tốt hơn.
Batch_size là số ảnh trong 1 lần duyệt qua epoch.
Trong lúc thực thi chương trình có xảy ra một vấn đề nhỏ đó là folder .ipynb_checkpoints xuất hiện liên tục trong các class có sẵn, do đó nhóm có tham khảo từ **StackOverFLow** để xóa thư mục trên nhằm các ảnh được gán đúng label:
rm -rf `find -type d -name .ipynb_checkpoints`

2.4.3. Tạo tập generator, dùng để huấn luyện dữ liệu:

```
data_generator = ImageDataGenerator(
 validation_split=0.2,
 rotation_range=20,
 width_shift_range=0.2,
 height_shift_range=0.2,
 rescale=1./255,
 shear_range=0.2,
 zoom_range=0.2,
 horizontal_flip=True,
 brightness_range=[0.8, 1.2],
 channel_shift_range=0.2,
 vertical_flip=True,
 fill_mode='nearest')
```

Bảng 2. Hàm tăng cường tập dữ liệu

- validation_split=0.2, : Tỉ lệ phần trăm dữ liệu được sử dụng cho validation set (tách từ dữ liệu huấn luyện). Khi có 500 ảnh trong tập data input thì ảnh để train sẽ là 400 và 100 ảnh dùng cho valid
- rotation_range=20, : Góc quay ảnh (degrees)
- width_shift_range=0.2, : Phạm vi dịch chuyển theo chiều ngang (tỷ lệ của bức ảnh)
- height_shift_range=0.2, : Phạm vi dịch chuyển theo chiều dọc (tỷ lệ của bức ảnh)
- rescale=1./255, : Tỉ lệ giảm độ sáng theo pixel
- shear_range=0.2, : Góc cắt (shear) ảnh

- zoom_range=0.2, : Tỉ lệ thu phóng ngẫu nhiên
- horizontal_flip=True, : Lật ngang ảnh theo chiều ngang
- brightness_range=[0.8, 1.2], : Phạm vi độ sáng của ảnh
- channel_shift_range=0.2, : Phạm vi dịch chuyển kênh màu (RGB) theo chiều ngang
- vertical_flip=True, : Lật dọc ảnh
- fill_mode='nearest') : trong lúc biến đổi hình ảnh, sẽ xuất hiện các pixel trống và chúng được điền theo pixel gần nhất

➔ Các bước trên nhằm tạo độ đa dạng cho các ảnh dùng cho việc train, từ một ảnh với các góc độ, ánh sáng, độ rộng khác nhau thì mô hình vẫn nhận dạng được hình ảnh được nạo vào.

2.4.4. Tạo tập huấn luyện:

Tập train_generator có tác dụng tạo ra dữ liệu hình ảnh cho các batch, có vai trò quan trọng trong việc đảm bảo dữ liệu trong quá trình huấn luyện. Việc tạo ra tập kiểm tra(valid) tương tự.

```
train_generator = data_generator.flow_from_directory('/content/train',
target_size=im_shape, shuffle=True,
seed=seed,class_mode='categorical', batch_size=BATCH_SIZE,
subset="training")
val_data_generator = ImageDataGenerator(rescale=1./255,
validation_split=0.2)
validation_generator =
val_data_generator.flow_from_directory('/content/train',
target_size=im_shape, shuffle=False,
seed=seed,class_mode='categorical', batch_size=BATCH_SIZE,
subset="validation

nb_train_samples = train_generator.samples
nb_validation_samples = validation_generator.samples
```

Bảng 3. Tạo tập huấn luyện và tập kiểm tra

- tập train_generator có tác dụng tạo ra dữ liệu hình ảnh cho các batch, ở đây sử dụng size(250,250), xóa trộn ảnh để các batch luôn được ngẫu nhiên, class_mode categorical có sẽ biểu diễn label dưới dạng nhị phân theo số lớp, ví dụ như này: "Cat": [1, 0, 0], với mục đích (subset) là training

- tập validation được tạo ra bằng cách lấy 20% số ảnh từ tập train và chức năng như 1 tập kiểm tra

2.4.5. Khai báo mô hình huấn luyện:

Mô hình chính ở đây là CNN Sequential (Tuần tự) với các thông số phù hợp. Trong đó, hàm mất mát(loss function) là crossentropy và trình tối ưu hóa là Adam

```
model = Sequential()
model.add(Conv2D(20, kernel_size=(3, 3
activation='relu',
input_shape=(250,250,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(40, kernel_size=(3,3), activation='relu'
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.2
model.add(Dense(num_classes, activation='softmax'
model.summary()

model.compile(loss='categorical_crossentropy',
optimizer=Adam(),
metrics=['accuracy'])
```

Bảng 4. Xây dựng mô hình CNN

- Tạo một mô hình tuần tự (Sequential): Dòng này khởi tạo một mô hình neural network tuần tự, nghĩa là các lớp trong mô hình sẽ được thêm tuần tự theo thứ tự.

- Thêm một lớp Conv2D: Đây là lớp tích chập đầu tiên trong mô hình. Nó được sử dụng để học các đặc trưng không gian từ dữ liệu hình ảnh. Trong trường hợp này, có 20 bộ lọc (filters) được sử dụng, mỗi bộ lọc có kích thước 3x3 và hàm kích hoạt là 'relu'. Input_shape cho lớp đầu tiên được chỉ định là (250, 250, 3), nghĩa là kích thước của ảnh là 250x250 và có 3 kênh màu (RGB).

- Thêm lớp MaxPooling2D: Lớp này được sử dụng để giảm kích thước của đầu ra từ lớp Conv2D trước đó, giúp giảm chi phí tính toán và tránh overfitting.

- Thêm lớp Conv2D khác: Lớp này tương tự như lớp Conv2D trước, nhưng có 40 bộ lọc và không cần chỉ định input_shape vì nó sẽ sử dụng đầu ra của lớp MaxPooling2D trước đó làm đầu vào.

- Thêm lớp Flatten: Lớp này chuyển đổi đầu ra 2D hoặc 3D từ lớp trước đó thành một vector 1D để đưa vào các lớp fully connected (dense).

- Thêm lớp Dense: Lớp này là một lớp fully connected với 100 nơ-ron và hàm kích hoạt là 'relu'.

- Thêm lớp Dropout: Lớp này giúp giảm overfitting bằng cách ngẫu nhiên loại bỏ một phần các nơ-ron trong quá trình huấn luyện.

- Thêm lớp Dense cuối cùng: Đây là lớp đầu ra của mô hình với số lượng nơ-ron bằng số lượng lớp đầu vào và hàm kích hoạt là 'softmax' để tính toán xác suất của các lớp đầu ra.

- In bảng tóm tắt của mô hình: Dòng này in ra bảng tóm tắt của mô hình, bao gồm thông tin về các lớp, số lượng tham số, và kích thước đầu ra của mỗi lớp.

- Biên dịch mô hình: Dòng này biên dịch mô hình với hàm mất mát là 'categorical_crossentropy', tối ưu hóa bằng Adam optimizer và độ chính xác làm đánh giá mô hình.

2.4.6. Huấn luyện mô hình:

```
epochs = 15
checkpoint_callback = ModelCheckpoint(filepath='model.h5',
monitor='val_loss', save_best_only=True, verbose=1)
early_stopping_callback = EarlyStopping(monitor='val_loss',
patience=2, verbose=1)
callbacks_list = [checkpoint_callback, early_stopping_callback]
history=model.fit(
        train_generator,
steps_per_epoch=nb_train_samples//BATCH_SIZE,
      epochs=epochs,
      callbacks = callbacks_list,
      validation_data=validation_generator,
```

```
    verbose = 1,
    validation_steps=nb_validation_samples // BATCH_SIZE)
```

Bảng 5. Hàm huấn luyện mô hình

- Định nghĩa số epochs: Biến epochs là số lượng epoch (vòng lặp) mà mô hình sẽ được huấn luyện qua dữ liệu.

- Định nghĩa các callbacks:

- ModelCheckpoint: Callback này được sử dụng để lưu trạng thái của mô hình vào một tệp tin (model.h5) mỗi khi có sự cải thiện trong val_loss (hàm mất mát trên tập validation). save_best_only=True chỉ lưu trạng thái của mô hình khi nó có hiệu suất tốt nhất trên tập validation.
- EarlyStopping: Callback này được sử dụng để dừng quá trình huấn luyện sớm nếu val_loss không cải thiện sau một số lượng epoch (được chỉ định bởi tham số patience).
- Gộp các callbacks vào một list: Các callbacks được gộp vào một list để truyền vào phương thức fit của mô hình.

- Huấn luyện mô hình:

- fit: Phương thức này bắt đầu quá trình huấn luyện của mô hình.
- train_generator và validation_generator được sử dụng để cung cấp dữ liệu cho việc huấn luyện và validation.
- steps_per_epoch và validation_steps là số bước (batch) mà mỗi epoch sẽ chạy qua trên tập huấn luyện và tập validation.
- epochs là số lượng epoch mà mô hình sẽ được huấn luyện.
- callbacks là danh sách các callbacks được gọi trong quá trình huấn luyện.
- verbose là cách mà thông tin về quá trình huấn luyện được hiển thị (0: không hiển thị, 1: hiển thị thông tin chi tiết, 2: hiển thị thông tin tổng quan).

➔ Kết quả của quá trình huấn luyện sẽ được lưu vào biến history để có thể truy cập và phân tích sau này

3. ĐÁNH GIÁ HỆ THỐNG VÀ KẾT LUẬN

Để đánh giá hệ thống 1 cách trực quan và chính xác, nhóm em đã thêm các hàm vẽ đồ thị và tính toán để dễ dàng quan sát

3.1. Loss và Accuracy

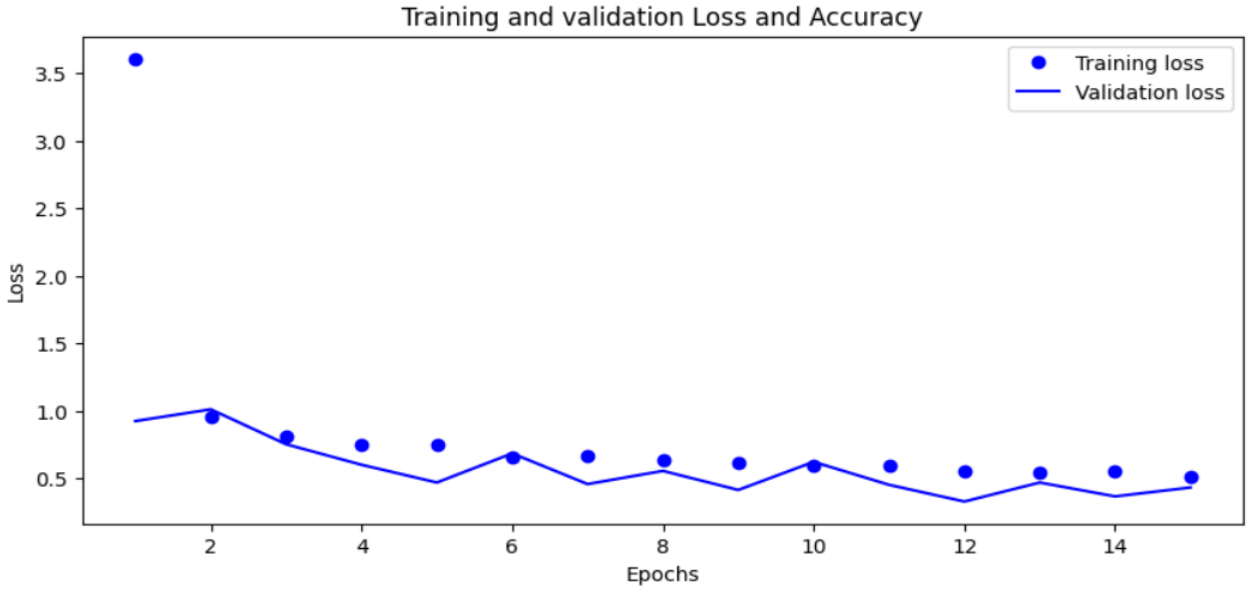
```
history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs_x = range(1, len(loss_values) + 1)
plt.figure(figsize=(10,10))
plt.subplot(2,1,1)
plt.plot(epochs_x, loss_values, 'bo', label='Training loss')
plt.plot(epochs_x, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation Loss and Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')

plt.legend()
plt.subplot(2,1,2)
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
plt.plot(epochs_x, acc_values, 'bo', label='Training acc')
plt.plot(epochs_x, val_acc_values, 'b', label='Validation acc')
plt.xlabel('Epochs')
plt.ylabel('Acc')
plt.legend()
plt.show()
```

Bảng 6. Hàm vẽ đồ thị Loss và Accuracy

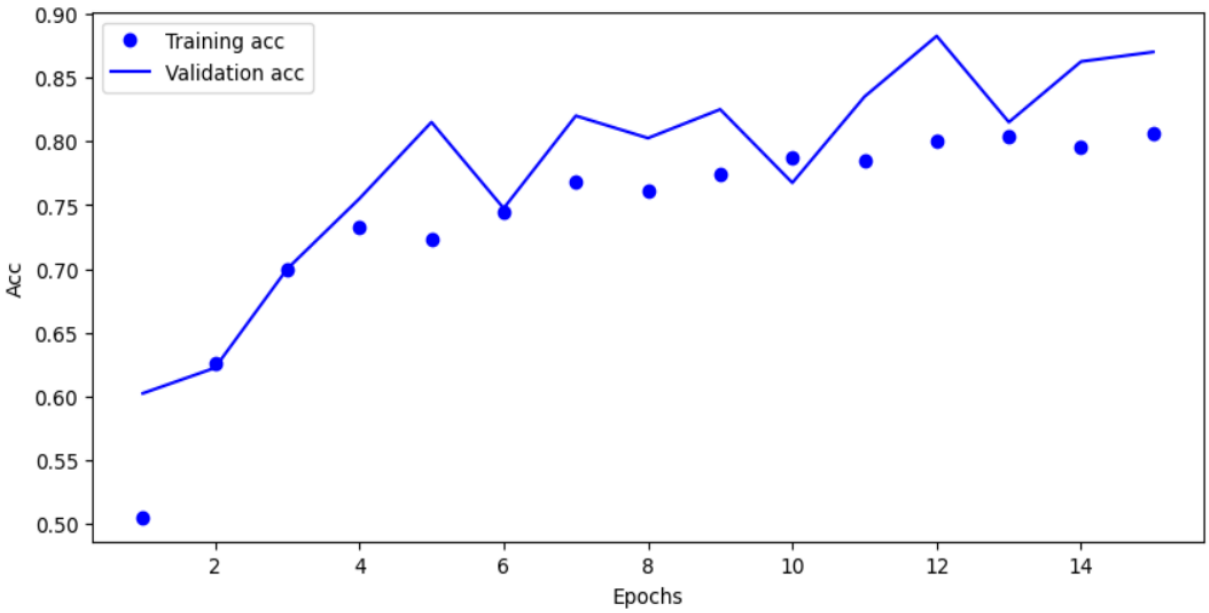
- Tạo biến history_dict: Biến này chứa các thông tin về hàm mất mát và độ chính xác từ quá trình huấn luyện mô hình, được lưu trong biến history.
- Vẽ đồ thị loss theo từng epoch:

- loss_values: Lấy danh sách các giá trị mất mát trên tập huấn luyện từ history_dict.
- val_loss_values: Lấy danh sách các giá trị mất mát trên tập validation từ history_dict.
- epochs_x: Tạo một list từ 1 đến số lượng epoch.
- Sử dụng thư viện matplotlib để vẽ đồ thị.

- Thiết lập subplot với 2 hàng và 1 cột.
- Sử dụng plt.plot để vẽ đồ thị mất mát của tập huấn luyện và tập validation.
- Thiết lập tiêu đề, tên trục và chú thích.
  - Vẽ đồ thị accuracy theo từng epoch:
- Tương tự như phần vẽ đồ thị loss, lấy các giá trị độ chính xác từ history_dict.
- Sử dụng plt.plot để vẽ đồ thị độ chính xác của tập huấn luyện và tập validation.
- Thiết lập tiêu đề, tên trục và chú thích.
- Cuối cùng là sử dụng plt.show() để hiển thị các đồ thị đã vẽ



Hình 4 – Đồ thị biểu diễn hàm Loss theo từng lần lặp



Hình 5 – đồ thị biểu diễn Accuracy theo từng lần lặp

3.2. Confusion Matrix

```
import itertools
from sklearn.metrics import classification_report, confusion_matrix
def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion
matrix', cmap=plt.cm.Blues):
    plt.figure(figsize=(10,10))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=2)
        cm[np.isnan(cm)] = 0.0
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
```

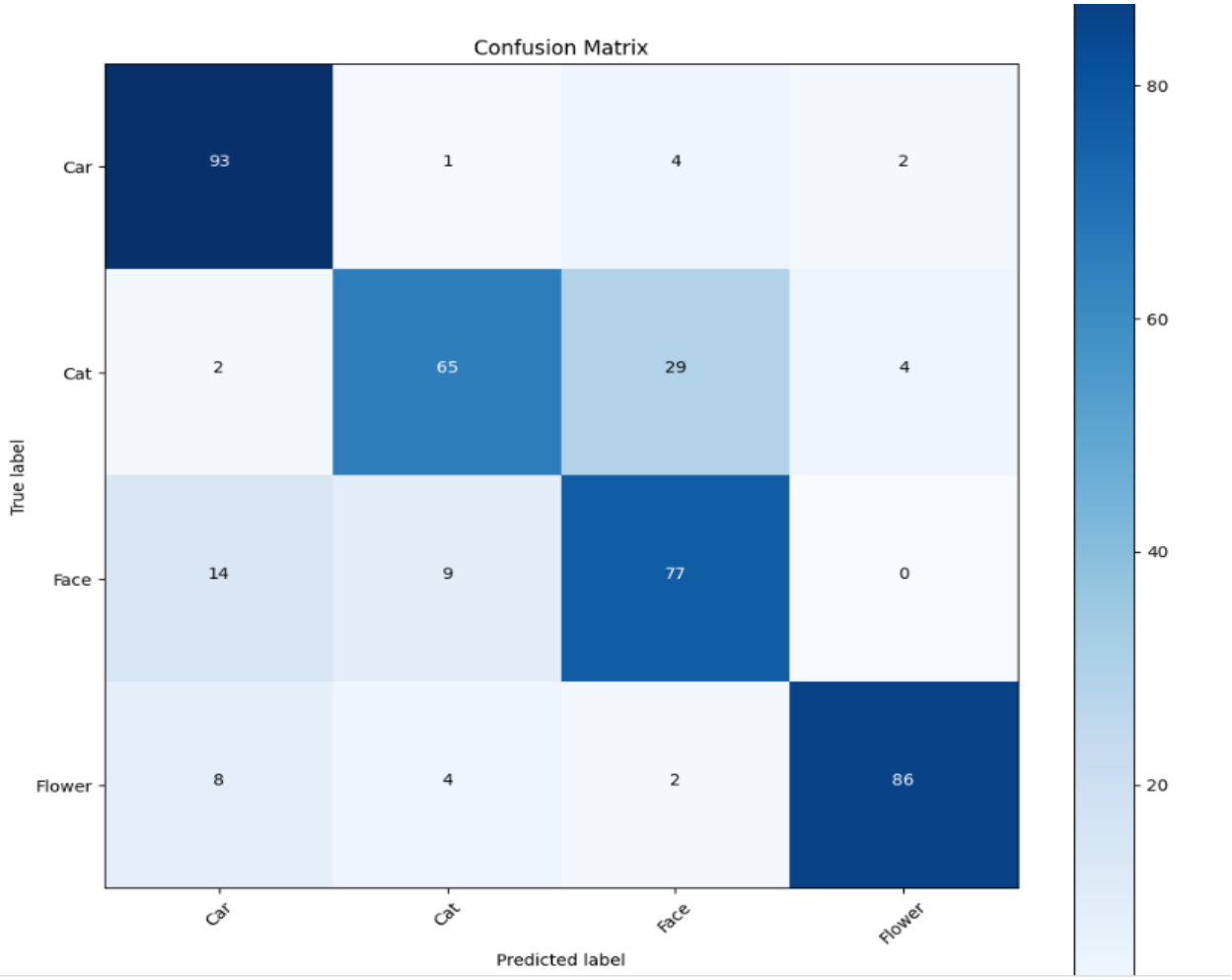
```
    plt.xlabel('Predicted label')

def evaluate_model(model, data_generator, classes):
    predictions = model.predict_generator(data_generator)
    predicted_labels = np.argmax(predictions, axis=1)
    true_labels = data_generator.classes
    cm = confusion_matrix(true_labels, predicted_labels)
    print("Confusion Matrix:")
    print(cm)

plot_confusion_matrix
    plot_confusion_matrix(cm, classes, normalize=False, title='Confusion Matrix')
    print('\nClassification Report:')
    print(classification_report(true_labels, predicted_labels,
target_names=classes))
evaluate_model(model, validation_generator, classes)
```

Bảng 7. Hàm tính toán các thông số và vẽ bảng confusion matrix

➔ Kết quả các thông số và confusion matrix như sau:

```
Classification Report:
              precision    recall  f1-score   support

         Car       0.79      0.93      0.86       100
         Cat       0.82      0.65      0.73       100
        Face       0.69      0.77      0.73       100
      Flower       0.93      0.86      0.90       100

    accuracy                           0.80       400
   macro avg       0.81      0.80      0.80       400
weighted avg       0.81      0.80      0.80       400
```

Hình 6 – các thông số classification



Hình 7 – Bảng confusion matrix

3.3. Nạp ảnh đã dự đoán vào folder và xuất ảnh kiểm tra

```
test_dir = '/content/test'
test_images = [os.path.join(test_dir, img) for img in os.listdir(test_dir)]

display_size = (250, 250)
output_dir = '/content/o5'
os.makedirs(output_dir, exist_ok=True)

for i in range(0, len(test_images), 4):
    fig, axes = plt.subplots(2, 2, figsize=(10, 10))
    for j, img_path in enumerate(test_images[i:i+4]):
        img = image.load_img(img_path, target_size=display_size)
```

```
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array /= 255.0  # Chuẩn hóa giá trị pixel về khoảng [0, 1]
        predictions = model.predict(img_array)
        predicted_label = np.argmax(predictions, axis=1)
        predicted_class = [k for k, v in train_generator.class_indices.items() if v
== predicted_label[0]][0]
        label_dir = os.path.join(output_dir, predicted_class)
        os.makedirs(label_dir, exist_ok=True)
        shutil.copy(img_path, label_dir)
        img = Image.open(img_path)
        img_array = np.array(img.resize(display_size)) / 255.0  # Resize ảnh và
chuẩn hóa giá trị pixel
        axes[j // 2, j % 2].imshow(img_array)
        axes[j // 2, j % 2].set_title(f"Predicted Label: {predicted_class}")
    plt.show()
```

Bảng 8. Hàm xuất ảnh vào từng folder riêng biệt có chứa nhãn

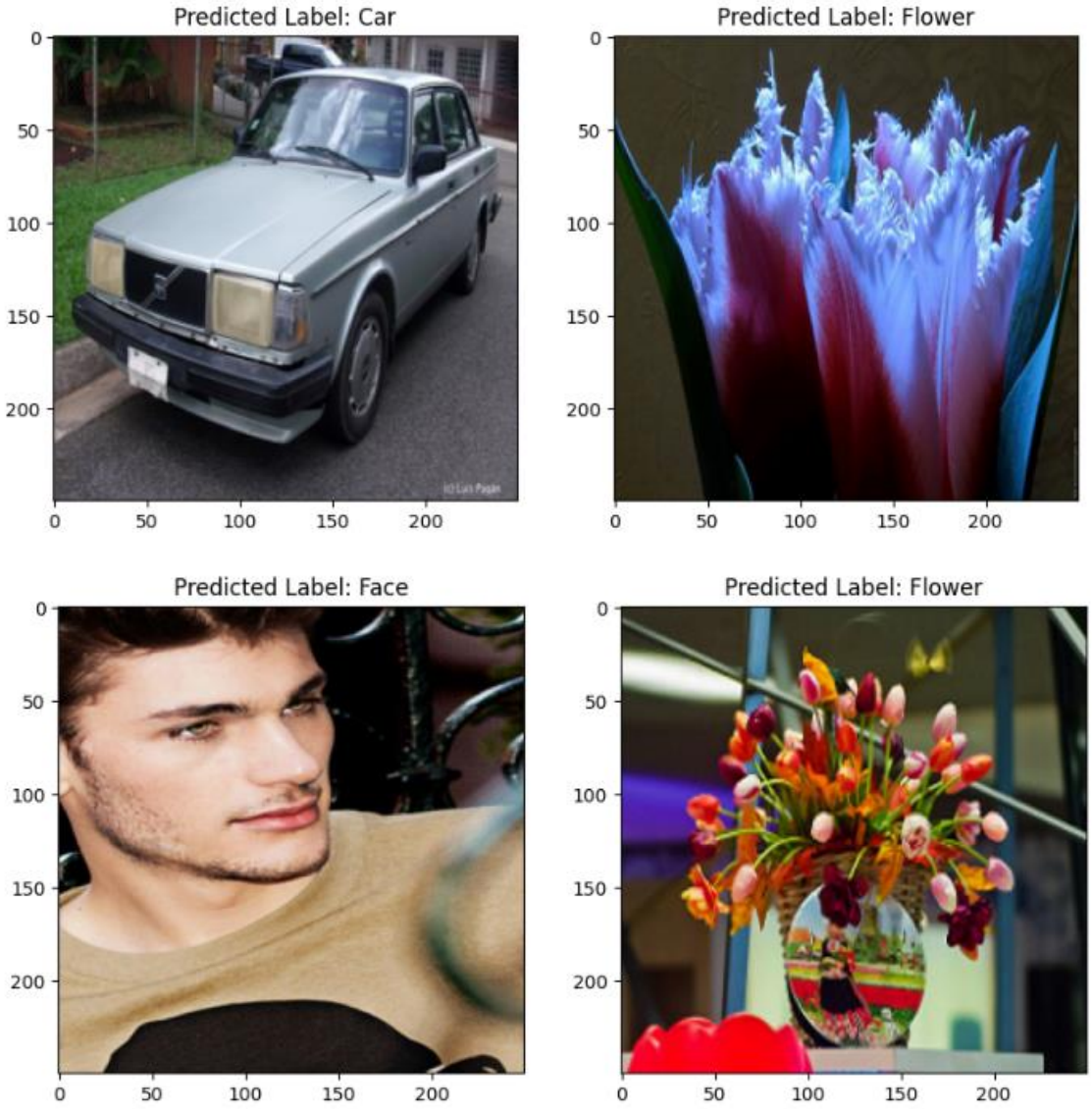
- Đầu tiên, data cho file test được xác định qua đường dẫn đến thư mục chứa ảnh test:

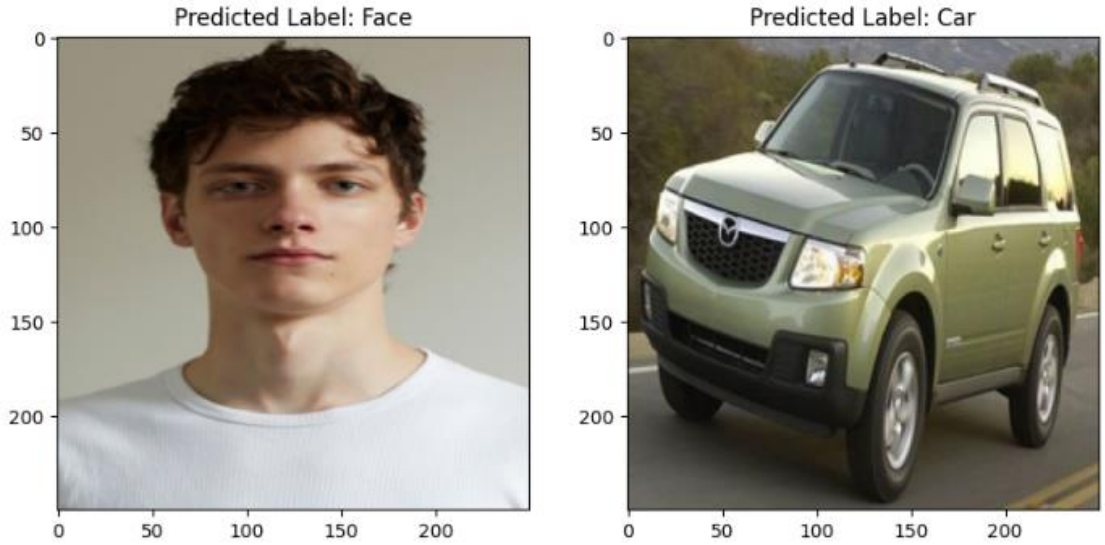
- Biến test_dir chứa đường dẫn tới thư mục chứa các ảnh cần dự đoán.
  - Tiếp theo,tạo một list chứa đường dẫn đầy đủ đến từng ảnh trong thư mục test:
- test_images là một list chứa đường dẫn đến từng ảnh trong thư mục test.
  - Xác định kích thước mới cho việc hiển thị:
- Biến display_size xác định kích thước ảnh sẽ được hiển thị trên biểu đồ.
  - Khởi tạo một thư mục để lưu các ảnh dự đoán, đây là bước để xác định các ảnh có cùng nội dung sẽ được nằm chung vào một folder:
- Biến output_dir chứa đường dẫn tới thư mục lưu các ảnh dự đoán.
- Hàm os.makedirs được sử dụng để tạo thư mục nếu chưa tồn tại.
  - Sau đó, duyệt qua từng ảnh trong tập test:
- Sử dụng vòng lặp để duyệt qua từng ảnh trong test_images theo batch có kích thước 4.
- Đối với mỗi batch, tạo một biểu đồ có 2 hàng và 2 cột để hiển thị 4 ảnh.
  - Dự đoán nhãn cho ảnh và lưu vào thư mục tương ứng:
- Đọc và chuẩn bị ảnh để dự đoán, sau đó sử dụng mô hình đã được huấn luyên ở trên để dự đoán nhãn.

- Dự đoán được giải mã từ class indices và sử dụng để tạo thư mục con tương ứng nếu chưa tồn tại.
- Ảnh được sao chép vào thư mục của nhãn dự đoán.
- Dùng matplotlib để hiển thị ảnh và nhãn dự đoán tương ứng trên biểu đồ.

Hình 8 – Output gồm các folder chứa ảnh được tạo ra

Một số kết quả mô phỏng :





Hình 9 – Một số ví dụ về ảnh và dự đoán nhãn của ảnh đó

Từ các đồ thị và ví dụ trên, có thể thấy hệ thống đã hiện thực được đúng theo mô tả đề tài và đã làm khá ổn việc phân loại hình ảnh không có nhãn. Khi tiến hành kiểm tra với tập validation gồm 400 ảnh với mỗi chủ đề chiếm 100 ảnh, ta thu được kết quả là các folder chứa các ảnh có cùng chung một nhãn với độ chính xác cuối cùng của tập validation là 0.88 và sự mất mát loss là 0.52 chỉ với sau 15 lần lặp.

3.4. Kết luận đề tài hiện tại

Tuy đề tài hiện tại còn nhiều yếu điểm và thực sự chưa hoàn chỉnh, chưa đủ phức tạp để có thể được coi là một bài toán Machine Learning đáng để nghiên cứu và có khả năng ứng dụng, đề tài và mô hình nhóm đã phát triển tuy có phần đơn giản, song nhìn chung nhóm cũng đã hoàn thành được mục tiêu đề ra của đề tài: Phát triển mô hình CNN có khả năng phân loại và xuất ra các ảnh cùng nhãn vào chung một tệp.

4. CÁC NGUỒN THAM KHẢO

Convolutional Neural Network (CNN). có sẵn tại: https://www.tensorflow.org/tutorials/images/cnn