BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM

KHOA CƠ KHÍ CHẾ TẠO MÁY

MÔN HỌC: TRÍ TUỆ NHÂN TẠO

BÁO CÁO CUỐI KỲ

XÂY DỰNG MÔ HÌNH NHẬN DIỆN LOÀI HOA BẰNG THUẬT TOÁN CNN

GVHD: PGS. TS Nguyễn Trường Thịnh

SVTH: Bùi Chí Cường

MSSV: 20146167

Mã lớp học: ARIN337629

Thành phố Hồ Chí Minh, Tháng 5 năm 2022

NHẬN XÉT CỦA GIẢNG VIÊN

……
……
……
……
……
……
……
……
……
……...

Điểm: ……..

KÝ TÊN

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN

1.1 Giới thiệu đề tài

Trong thời đại mà công nghệ ngày càng phát triển vượt bậc như hiện nay thì đề tài "tự động nhận diện và phân loại" hoa là một trong những đề tài quan trọng trong lĩnh vực trí tuệ nhân tạo. Đề tài này nhằm xây dựng một hệ thống có khả năng nhận diện và phân loại các loại hoa từ hình ảnh sử dụng thuật toán CNN (Convolutional Neural Network).

Mạng CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo được thiết kế đặc biệt để xử lý và phân loại dữ liệu hình ảnh và video. Nó được khám phá và phát triển dựa trên các cơ sở lý thuyết về thị giác máy tính và mô phỏng cách thức hoạt động của hệ thống nơ-ron thần kinh của con người.

Mạng CNN có cấu trúc chính gồm các lớp convolutional (lớp tích chập), pooling (lớp gộp), và fully connected layers (lớp kết nối đầy đủ). Mạng CNN đã chứng minh khả năng xuất sắc trong việc nhận diện và phân loại hình ảnh. Nó đã được áp dụng rộng rãi trong nhiều lĩnh vực như nhận diện khuôn mặt, nhận diện loài hoa, nhận dạng đối tượng, phân loại hình ảnh y học, và nhiều ứng dụng thị giác máy tính khác.

Với sự phát triển của các tập dữ liệu lớn và khả năng tính toán mạnh mẽ, CNN có thể huấn luyện trên hàng triệu hình ảnh để tạo ra một mô hình có độ chính xác cao và khả năng tổng quát hóa tốt. Cùng với khả năng học tập tự động từ dữ liệu và khả năng xử lý thông tin không gian của hình ảnh, mạng CNN đã trở thành công cụ quan trọng trong phân tích hình ảnh và trí tuệ nhân tạo.

Trong bài báo cáo này, em sẽ tiến hành xây dựng một mô hình thuật toán với mục đích là nhận diện và phân loại loài bằng thuật toán Convolutional Neural Network (CNN) và tạo ra một giao diện GUI để có thể nhận diện loài hoa trực tiếp bằng camera hoặc thông qua dữ liệu ảnh.

1.2 Các nội dung thực hiện

Chọn loại hoa để thực hiện đối với đề tài này và tìm hiểu ứng dụng của mạng CNN trong việc nhận diện và phân loại loài hoa thông qua phân tích các chỉ số đặc trưng từ một tấm ảnh.

Thu thập dữ liệu: Sau khi chọn được loài hoa mong muốn thì em sẽ đi thu thập dữ liệu về loài hoa đó. Tiếp đến xác định các yếu tố ảnh hưởng đến việc xác định loài hoa như: màu sắc, hình dạng, chiều dài, chiều rộng,...và thu thập một lượng dữ liệu với một dung lượng vừa đủ để có thể ứng dụng một cách linh hoạt những kiến thức đã học từ môn học để xử lý dữ liệu thu thập được.

Tiền xử lý dữ liệu: Tiến hành các bước tiền xử lý dữ liệu cần thiết, bao gồm loại đưa tất cả ảnh về một kích thước chuẩn, chuẩn hóa và mã hóa các biến cần thiết cho việc huấn luyện mô hình Machine Learning. Dán nhãn cho bộ ảnh và chia ra tập nhãn (labels) và tập ảnh (photos) rồi lưu lại. Cuối cùng, chia ra tập huấn luyện (train) và tập kiểm nghiệm (test) để chuẩn bị cho việc huấn luyện mô hình.

Xây dựng mô hình CNN: sau khi chọn được mô hình phù hợp, bắt đầu tiến hành xây dựng mô hình của thuật toán với các thông số thiết lập bắt buộc.

Đào tạo và đánh giá mô hình: Từ mô hình đã xây dựng ở trên, bắt đầu tiến hành huấn luyện mô hình CNN trên tập dữ liệu huấn luyện đã xử lý từ trước và đánh giá hiệu suất của mô hình chạy được từ tập dữ liệu thử nghiệm.

Đánh giá và so sánh kết quả: So sánh kết quả đạt được của các mô hình Machine Learning đã thực hiện và đánh giá tính khả thi và hiệu quả của mô hình đó trong việc phát hiện và nhận diện loài hoa.

Xây dựng giao diện đồ họa người dùng (GUI): Sau quá trình xây dựng thành công mô hình tiếp đến tiến hành xây dựng giao diện đồ họa người dùng (GUI) với việc sử dụng camera để nhận diện trực tiếp hoặc tải ảnh bất kỳ để nhận diện.

CHƯƠNG 2: QUÁ TRÌNH THỰC HIỆN

Sau đây là quá trình xây dựng mô hình nhận diện loài hoa:

- Gọi ra những thư viện cần thiết cho việc huấn luyện mô hình.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from google.colab import drive
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
import tensorflow as tf
from keras.models import Sequential
from keras.optimizers import Adam
from keras.losses import categorical_crossentropy
from keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten,
                         Dense, Dropout, BatchNormalization, LeakyReLU
from keras.layers.attention.multi_head_attention import activation
import seaborn as sns
import cv2
from os import listdir
from PIL import Image
from numpy.core.multiarray import asarray
from keras.utils import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

- Tạo đường dẫn tập ảnh dữ liệu cần cho đề tài từ drive vào biến ‘folder’.

- Sau đó, tạo hai danh sách (list) trống có tên lần lượt là photos và labels để lưu trữ các ảnh và nhãn tương ứng với từng ảnh.

- Trong vòng lặp ‘for’, dùng hàm ‘listdir’ để lấy ảnh chứa trong biến ‘folder’ đã khai báo ở trên. Lần lượt đọc tất cả các ảnh chứa trong ‘folder’ từng ảnh và gán nhãn tương ứng cho ảnh đó. Với ảnh bắt đầu lần lượt là 'Catharanthus', 'CrownlmperialI', 'Daisy', 'Hibiscus', 'Ixora', 'PeaceLily', 'Poinsettia', 'Rose', 'Sunflower', 'Tulip' thì gán nhãn lần lượt là các giá trị [0.0 ; 1.0 ; 2.0 ; 3.0 ; 4.0 ; 5.0 ; 6.0 ; 7.0 ; 8.0 ; 9.0] và 10.0 là cho các ảnh không có tên như trên. Sau đó, đọc ảnh bằng hàm ‘load_img()’, rồi thay đổi kích thước ảnh về 40x40 pixels bằng ‘target_size’ và chuyển thành mảng numpy bằng hàm ‘img_to_array()’.

- Thêm ảnh với các biến đã được gán ở trên lần lượt vào biến ‘photos’ và nhãn ‘labels’.

- Chuyển đổi 2 danh sách ‘photos’ và ‘labels’ thành các mãng numpy bằng hàm ‘asarray’.

- Dùng lệnh ‘print’ in ra kích thước của mảng ảnh (‘photos’) và mảng nhãn (‘labels’) bằng hàm ‘shape’.

```
folder = '/content/drive/MyDrive/Colab Notebooks/Final_Course_Homeworks/Flower_10_data/'
photos, labels = list(), list()
for file in listdir(folder):
  output = 10.0
  if file.startswith('Catharanthus'):
    output = 0.0
  if file.startswith('Crownlmperial'):
    output = 1.0
  if file.startswith('Daisy'):
    output = 2.0
  if file.startswith('Hibiscus'):
    output = 3.0
  if file.startswith('Ixora'):
    output = 4.0
  if file.startswith('PeaceLily'):
    output = 5.0
  if file.startswith('Poinsettia'):
    output = 6.0
  if file.startswith('Rose'):
    output = 7.0
  if file.startswith('Sunflower'):
    output = 8.0
  if file.startswith('Tulip'):
    output = 9.0

  photo = load_img(folder+file, target_size = (40,40))
  photo = img_to_array(photo)
  photos.append(photo)
  labels.append(output)
photos = asarray(photos)
labels = asarray(labels)
print(photos.shape, labels.shape)
```

- Lưu 2 tập ‘photos’ và ‘labels’ vào drive với đường dẫn xác định cụ thể như trên để có thể gọi ra sử dụng bất cứ khi nào cần cho việc huấn luyện model.

```
# Save data
np.save('/content/drive/MyDrive/Colab Notebooks/Final_Course_Homeworks/Flower_photos_1.npy', photos)
np.save('/content/drive/MyDrive/Colab Notebooks/Final_Course_Homeworks/Flower_labels_1.npy', labels)
```

- Gọi hai tập dữ liệu ‘photos’ và ‘labels’ đã lưu ở drive và gán vào 2 biến ‘photos’ và ‘labels’.

```
# Load data
photos = np.load('/content/drive/MyDrive/Colab Notebooks/Final_Course_Homeworks/Flower_photos_1.npy')
labels = np.load('/content/drive/MyDrive/Colab Notebooks/Final_Course_Homeworks/Flower_labels_1.npy')
```

- Sử dụng hàm ‘train_test_split’ để chia tập dữ liệu đã xử lý ở trên thành 2 tập train và test theo tỷ lệ 90:10, hệ số ‘random_state=10’ có tác dụng tạo lại 2 tập train và test giống nhau sau mỗi lần chạy.

- Dùng hàm ‘shape’ để xem kích thước của tập dữ liệu train và test vừa chia.

```
# split train data, test data
x_train, x_test, y_train, y_test = train_test_split(photos, labels, test_size=0.1, random_state=10)
```

```
x_train.shape
```

```
(4678, 40, 40, 3)
```

```
x_test.shape
```

```
(520, 40, 40, 3)
```

```
y_train.shape
```

```
(4678,)
```

```
y_test.shape
```

- Chuẩn hóa dữ liệu và chuyển đổi dạng nhãn thành dạng one-hot encoding. Với hàm ‘astype()’ dữ liệu được chuyển sang kiểu float32 và chia mỗi giá trị điểm ảnh trong phạm vi từ 0 đến 255 cho 255 để chuẩn hóa giá trị điểm ảnh về khoảng [0, 1]. Điều này giúp cho mô hình huấn luyện tốt hơn và dễ dàng hội tụ hơn. Với hàm ‘to_categorical()’ được sử dụng để chuyển đổi nhãn dưới dạng số thành one-hot encoding.

- Dùng hàm ‘shape’ để xem kích thước của 2 tập nhãn Y_train, Y_test sau khi được chuẩn hóa.

```
# Standardized data

x_train = x_train.astype('float32')/255
y_train = to_categorical(y_train, 11)

x_test = x_test.astype('float32')/255
y_test = to_categorical(y_test, 11)
```

```
y_train.shape
```

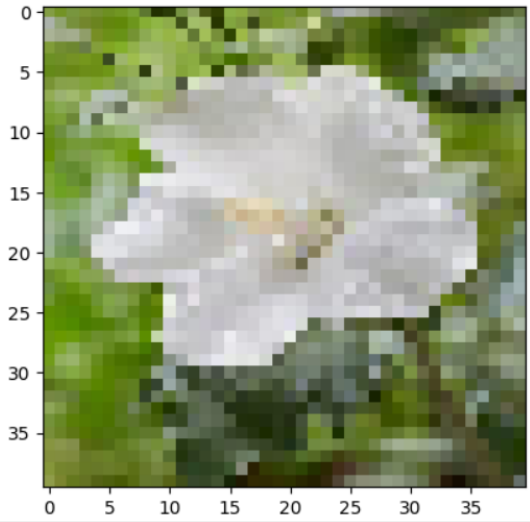
```
(4678, 11)
```

```
y_test.shape
```

```
(520, 11)
```

- In ra thử nghiệm một ảnh từ trong tập dữ liệu để kiểm tra xem dữ liệu đã được chuẩn hóa như mong muốn hay chưa. Nếu chưa thì tiến hành sửa lỗi, nếu dữ liệu đã phù hợp như dự định thì tiến hành xây dựng và huấn luyện mô hình.

```
img = x_train[1000]
plt.imshow(img)
plt.show()
print(y_train[1000])
```



- Khởi tạo xây dựng thuật toán CNN.

- Thêm một lớp Conv2D với input_shape=(40, 40, 3) để xử lý hình ảnh đầu vào. Lớp này có 128 bộ lọc (filters) có kích thước (3,3), strides=(1, 1) và padding='same'. Hàm kích hoạt (activation) được sử dụng là 'relu'.

- Tiếp theo, thêm lớp Conv2D với 128 bộ lọc, strides=1, padding='same' và activation='relu'.

- Thêm một lớp MaxPooling2D với pool_size=(2, 2) và strides=(2, 2) để giảm kích thước các đặc trưng đầu ra.

- Thêm một khối với hai lớp Conv2D với 256 bộ lọc, strides=1, padding='same' và activation='relu' và một lớp MaxPooling2D với pool_size=(2, 2) để giảm kích thước các đặc trưng đầu ra.

- Tiếp tục thêm hai khối với mỗi khối chứa ba lớp Conv2D với 512 bộ lọc, strides=1, padding='same' và activation='relu' và một lớp MaxPooling2D với pool_size=(2, 2) để giảm kích thước các đặc trưng đầu ra..

- Sau các lớp tích chập, sử dụng một lớp Flatten để chuyển đổi đặc trưng thành một vector 1 chiều.

- Tiếp theo, có các lớp Dense với 512, 256, và 11 đơn vị, tương ứng. Các lớp này sử dụng hàm kích hoạt 'relu' và cuối cùng là lớp Dense với hàm kích hoạt 'softmax' để đưa ra dự đoán phân loại cho 11 lớp (11 loại hoa)

- Dùng hàm ‘summary()’ để sẽ hiển thị tổng quan kiến trúc của mô hình mạng nơ-ron. Cụ thể, nó sẽ hiển thị các tầng (layers) của mô hình, kích thước đầu vào và đầu ra của từng tầng, số lượng tham số trong mỗi tầng, số lượng tham số cần được huấn luyện trong mô hình, tổng số tham số và kích thước đầu vào và đầu ra của toàn bộ mô hình.

- Hàm ‘compile()’ để tiến hành biên dịch chương trình với các thông số hàm mất mát loss='categorical_crossentropy', thuật toán tự thích ứng ‘Adam’ và tỉ lệ học tập 0.0001 optimizer=Adam(learning_rate=0.0001), chỉ số đánh giá hiệu suất mô hình huấn luyện metrics=['accuracy'].

```
model = Sequential()

model.add(Conv2D(128, (3, 3), strides=(1,1), padding = 'same', activation = 'relu',
                 input_shape=(40, 40, 3)))
model.add(Conv2D(128, (3, 3),strides=(1,1), padding = 'same',  activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3), strides=(1,1), padding = 'same', activation = 'relu'))
model.add(Conv2D(256, (3, 3), strides=(1,1), padding = 'same', activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(512, (3, 3), strides=(1,1), padding = 'same', activation = 'relu'))
model.add(Conv2D(512, (3, 3), strides=(1,1), padding = 'same', activation = 'relu'))
model.add(Conv2D(512, (3, 3), strides=(1,1), padding = 'same', activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(512, (3, 3), strides=(1,1), padding = 'same', activation = 'relu'))
model.add(Conv2D(512, (3, 3), strides=(1,1), padding = 'same', activation = 'relu'))
model.add(Conv2D(512, (3, 3), strides=(1,1), padding = 'same', activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(11, activation='softmax'))

model.summary()
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.0001), metrics=['accuracy'])
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_20 (Conv2D)          (None, 40, 40, 128)       3584

 conv2d_21 (Conv2D)          (None, 40, 40, 128)       147584

 max_pooling2d_8 (MaxPooling  (None, 20, 20, 128)      0
 2D)

 conv2d_22 (Conv2D)          (None, 20, 20, 256)       295168

 conv2d_23 (Conv2D)          (None, 20, 20, 256)       590080

 max_pooling2d_9 (MaxPooling  (None, 10, 10, 256)      0
 2D)

 conv2d_24 (Conv2D)          (None, 10, 10, 512)       1180160

 conv2d_25 (Conv2D)          (None, 10, 10, 512)       2359808

 conv2d_26 (Conv2D)          (None, 10, 10, 512)       2359808

 max_pooling2d_10 (MaxPoolin  (None, 5, 5, 512)        0
 g2D)

 conv2d_27 (Conv2D)          (None, 5, 5, 512)         2359808

 conv2d_28 (Conv2D)          (None, 5, 5, 512)         2359808

 conv2d_29 (Conv2D)          (None, 5, 5, 512)         2359808

 max_pooling2d_11 (MaxPoolin  (None, 2, 2, 512)        0
 g2D)

 flatten_2 (Flatten)         (None, 2048)              0

 dense_6 (Dense)             (None, 512)               1049088

 dense_7 (Dense)             (None, 256)               131328

 dense_8 (Dense)             (None, 11)                2827

=================================================================
Total params: 15,198,859
Trainable params: 15,198,859
Non-trainable params: 0
```

- Tiếp theo đến bước làm giàu hóa dữ liệu, sử dụng ImageDataGenerator để tạo ra các phiên bản biến đổi của dữ liệu huấn luyện. Các biến đổi bao gồm:

rotation_range=10: Quay ảnh trong khoảng -10 đến +10 độ.

width_shift_range=0.05: Dịch chuyển ảnh theo chiều ngang trong khoảng -5% đến +5% của chiều rộng.

height_shift_range=0.05: Dịch chuyển ảnh theo chiều dọc trong khoảng -5% đến +5% của chiều cao.

shear_range=0.05: Xoắn ảnh trong khoảng -5% đến +5%.

zoom_range=0.05: Phóng to hoặc thu nhỏ ảnh trong khoảng -5% đến +5%.

horizontal_flip=True: Lật ngang ảnh.

fill_mode='nearest': Điền giá trị gần nhất cho các pixel sau khi thực hiện biến đổi.

```
aug = ImageDataGenerator(rotation_range=10,
                         width_shift_range=0.05,
                         height_shift_range=0.05,
                         shear_range=0.05,
                         zoom_range=0.05,
                         horizontal_flip=True,
                         fill_mode='nearest')
```

- Tiến hành quá trình huấn luyện mô hình. Sử dụng phương thức fit_generator để huấn luyện từng batch dữ liệu. Các tham số được sử dụng bao gồm:

aug.flow(X_train, Y_train, batch_size=100): Dữ liệu huấn luyện được truyền vào dưới dạng generator từ phương thức flow của đối tượng aug (ImageDataGenerator). Batch size là 100, tức là mỗi lần huấn luyện sẽ sử dụng 100 mẫu dữ liệu.

epochs=35: Số lượng epoch, tức là số lần huấn luyện trên toàn bộ dữ liệu huấn luyện.

validation_data=aug.flow(X_test,Y_test, batch_size=100): Dữ liệu kiểm tra được truyền vào dưới dạng generator từ phương thức flow của đối tượng aug. Batch size là 100.

```
train = model.fit_generator(aug.flow(x_train, y_train, batch_size=100),
                            epochs=35, validation_data=aug.flow(x_test,y_test, batch_size=100))
```

- Cuối cùng, độ chính xác trên tập train là 0.939, còn trên tập test là 0.836.

```
Epoch 27/35
47/47 [==============================] - 11s 226ms/step - loss: 0.2998 - accuracy: 0.8942 - val_loss: 0.7272 - val_accuracy: 0.7942
Epoch 28/35
47/47 [==============================] - 7s 149ms/step - loss: 0.3082 - accuracy: 0.8948 - val_loss: 0.7158 - val_accuracy: 0.7962
Epoch 29/35
47/47 [==============================] - 7s 152ms/step - loss: 0.2894 - accuracy: 0.9015 - val_loss: 0.7118 - val_accuracy: 0.8173
Epoch 30/35
47/47 [==============================] - 7s 154ms/step - loss: 0.2211 - accuracy: 0.9265 - val_loss: 0.6702 - val_accuracy: 0.8250
Epoch 31/35
47/47 [==============================] - 7s 149ms/step - loss: 0.2035 - accuracy: 0.9322 - val_loss: 0.6680 - val_accuracy: 0.8231
Epoch 32/35
47/47 [==============================] - 7s 153ms/step - loss: 0.1849 - accuracy: 0.9348 - val_loss: 0.7356 - val_accuracy: 0.8385
Epoch 33/35
47/47 [==============================] - 7s 149ms/step - loss: 0.1725 - accuracy: 0.9384 - val_loss: 0.6826 - val_accuracy: 0.8577
Epoch 34/35
47/47 [==============================] - 7s 151ms/step - loss: 0.2019 - accuracy: 0.9254 - val_loss: 0.6502 - val_accuracy: 0.8308
Epoch 35/35
47/47 [==============================] - 7s 147ms/step - loss: 0.1677 - accuracy: 0.9397 - val_loss: 0.6547 - val_accuracy: 0.8365
```

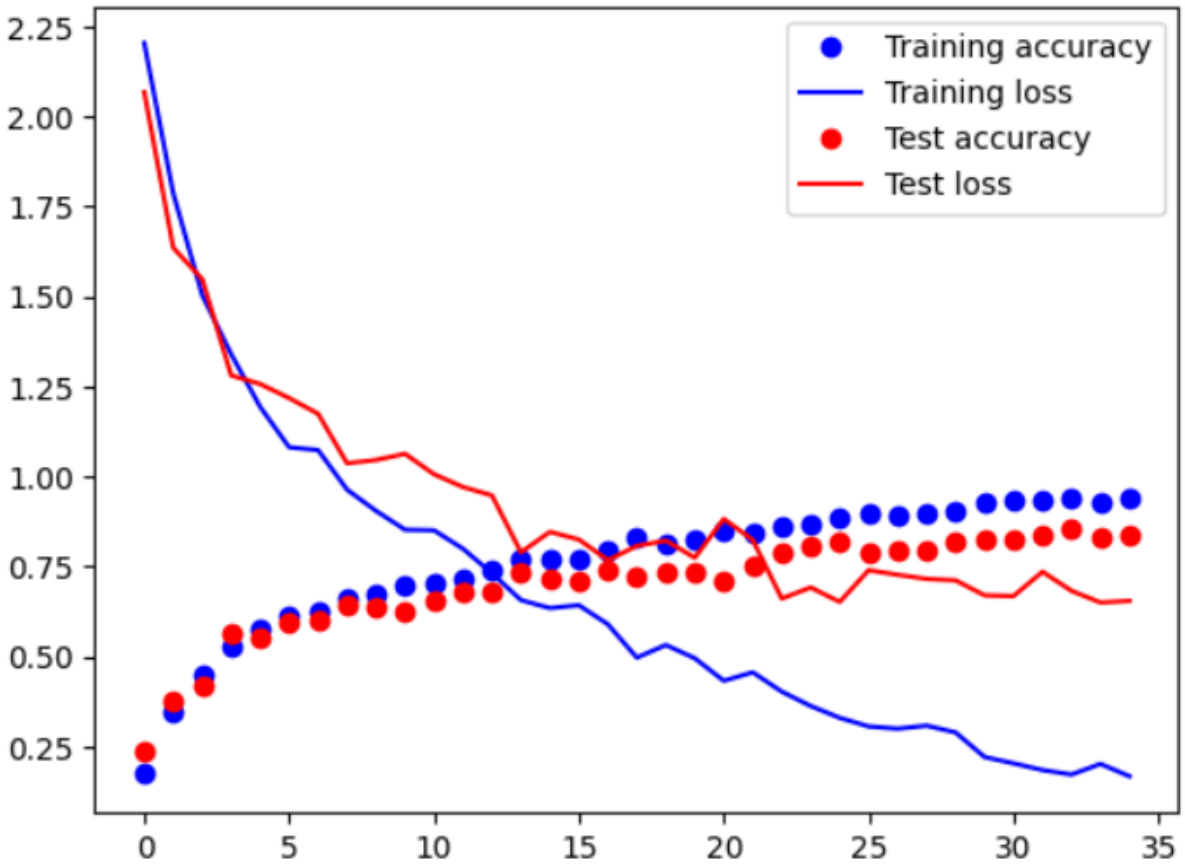
- Mô hình được lưu vào drive dưới dạng file .h5 với đường dẫn '/content/drive/MyDrive/Colab Notebooks/Final_Course_Homeworks/Predict_flower_model_f2.h5'

```
# Save model
model.save('/content/drive/MyDrive/Colab Notebooks/Final_Course_Homeworks/Predict_flower_model_f2.h5')
```

- Đoạn chương trình này, để đánh độ chính xác của mô hình vừa được huấn luyện bằng hàm 'evaluate'.

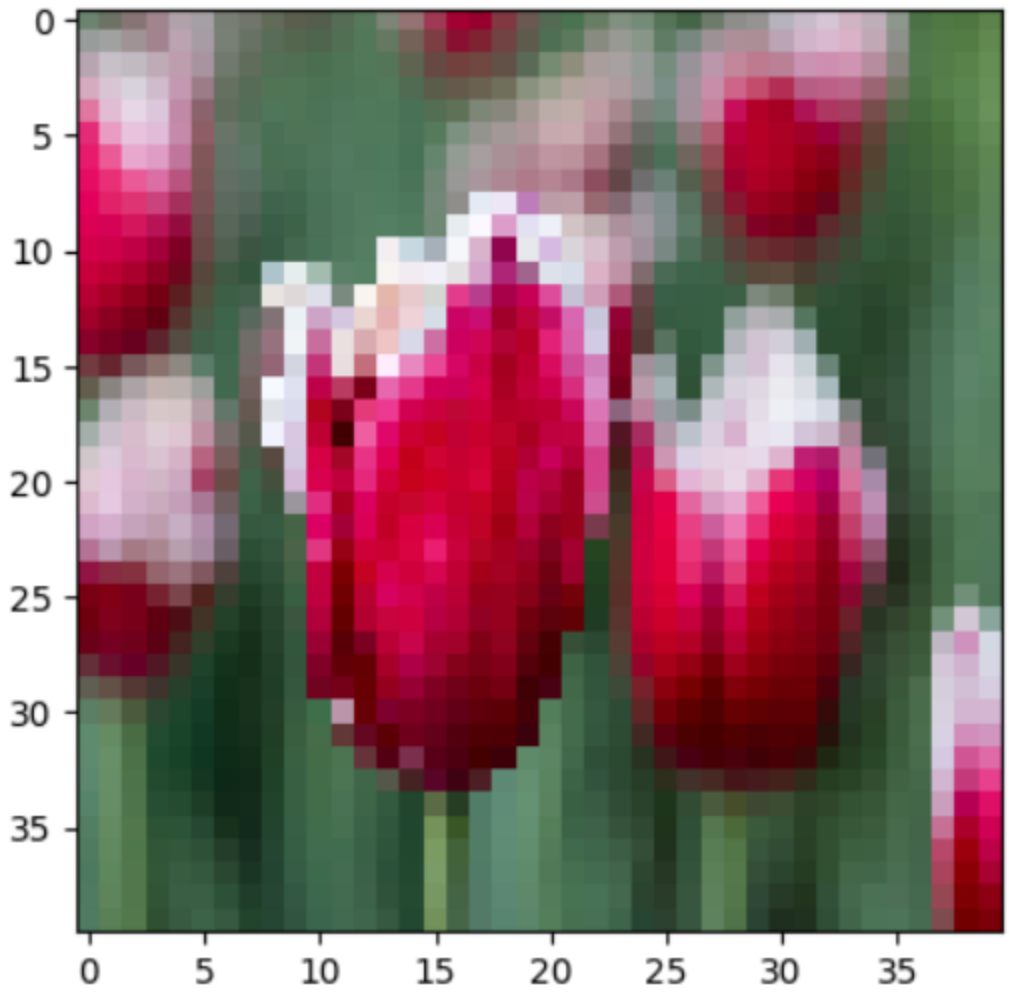
- Sau đó, em lấy giá trị độ chính xác và độ tổn thất của quá trình huấn luyện trên từng epoch bằng cách lấy thông tin lưu trữ trong hàm 'train.history'. Rồi vẽ đồ thị biểu diễn độ chính xác và độ tổn thất của quá trình huấn luyện theo epoch để dễ dàng theo dõi và đánh giá hiệu suất của mô hình. Với đường màu đỏ nét chấm biểu diễn độ chính xác, và nét liền biểu diễn độ mất mát trên tập thử nghiệm. Với đường màu xanh dương nét chấm biểu diễn độ chính xác, và nét liền biểu diễn độ mất mát trên tập thử nghiệm

```
test = model.evaluate(x_test, y_test, verbose=0)
print('Test loss: ', test[0])
print('Test accuracy', test[1])

accuracy=train.history['accuracy']
loss=train.history['loss']

test_accuracy=train.history['val_accuracy']
test_loss=train.history['val_loss']

epochs=range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Training accuracy',color='blue')
plt.plot(epochs, loss, 'b', label='Training loss',color='blue')

plt.plot(epochs, test_accuracy, 'bo', label='Test accuracy',color='red')
plt.plot(epochs, test_loss, 'b', label='Test loss',color='red')

plt.legend()
plt.show()
```

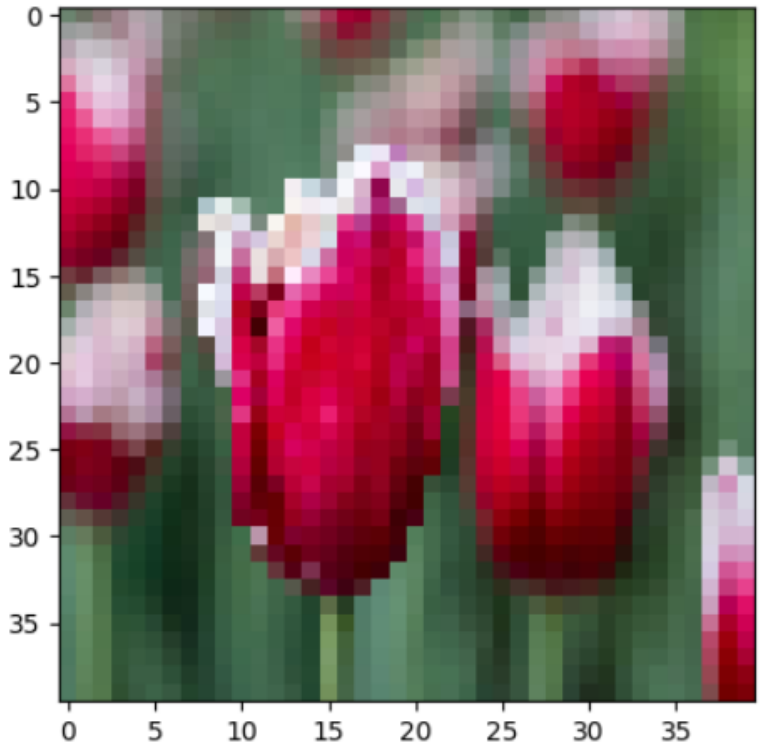


- Để kiểm nghiệm độ chính xác của mô hình,ở đoạn chương trình dưới em tải ảnh từ tập test và nhãn tương ứng để chuẩn bị dữ liệu đánh giá mô hình. Với ảnh là hoa ‘tulip’ được gán vào output là ‘9.0’.

```
img = x_test[121]
plt.imshow(img)
plt.show()
print(y_test[121])
```




```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
```




```
[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
```

```
x_pred = x_test[121].reshape(1,40,40,3)
x_pred=x_pred.astype('float32')/255
matrix = np.argmax(model.predict(x_pred))
class_names = ["Catharanthus", "Crownlmperial", "Daisy", "Hibiscus", "Ixora",
               "PeaceLily", "Poinsettia", "Rose", "Sunflower", "Tulip", "Dont know"]

class_predict = matrix
print("Predict flower: ", class_names[class_predict])
```

```
1/1 [==============================] - 0s 76ms/step
Predict flower:  Tulip
```

Tiếp theo, dưới đây là quá trình xây dựng giao diện đồ họa người dùng (GUI):

- Import những thư viện cần thiết cho việc xây dựng giao diện.

```
# Import libraries
import tkinter as tk
import numpy as np
import cv2
import time
from PIL import ImageTk
from PIL import Image
from keras.models import load_model
from tkinter import filedialog
```

- Gọi mô hình đã huấn luyện và đã được lưu lại.

- Tạo kích thước cố định cho ảnh đầu vào.

- Tạo danh sách các nhãn cho việc dự đoán.

```
# Load mô hình từ file .h5
model = load_model("Predict_flower_model_f.h5")

# Kích thước mong muốn của ảnh đầu vào
input_shape = (40, 40)

# Danh sách nhãn
labels = ['Catharanthus','CrownImperial','Daisy','Hibiscus','Ixora' ,
          'PeaceLily','Poinsettia', 'Rose','Sunflower','Tulip','No understand']
```

- Tiếp đến, tạo hàm “recognize_flower()” để xử lý ảnh đầu vào, dùng mô hình dự đoán và tạo ra nhãn kết quả.

```
def recognize_flower(image):
    # Thay đổi kích thước ảnh
    resized_image = image.resize(input_shape)
    # Chuyển đổi ảnh sang RGB
    rgb_image = resized_image.convert('RGB')
    # Chuẩn hóa dữ liệu đầu vào và chuyển sang numpy array
    flower_image = np.array(rgb_image) / 255.0
    # Thêm một chiều để phù hợp với input_shape của mô hình
    flower_image = np.expand_dims(flower_image, axis=0)
    # Dự đoán nhãn hoa
    prediction = model.predict(flower_image)
    predicted_label = np.argmax(prediction)
    return predicted_label
```

- Tạo hàm “detect_from_camera()” để nhận diện hoa trực tiếp từ camera.

```python
def detect_from_camera():
    # Mở kết nối với camera
    cap = cv2.VideoCapture(0)

    while True:
        # Đọc khung ảnh từ camera
        ret, frame = cap.read()

        # Tính toán tọa độ cho khung hiển thị
        frame_height, frame_width = frame.shape[:2]
        top_left = ((frame_width - 250) // 2, (frame_height - 250) // 2)
        bottom_right = (top_left[0] + 250, top_left[1] + 250)

        # Vị trí hoa
        text = '+'
        font_scale = 0.7
        font_thickness = 2
        font = cv2.FONT_HERSHEY_SIMPLEX
        text_size, _ = cv2.getTextSize(text, font, font_scale, font_thickness)
        text_width = text_size[0]
        text_height = text_size[1]
        text_x = top_left[0] + (350 - text_width) // 2
        text_y = top_left[1] + (350 - text_height) // 2
```

```
    # Chuyển đổi frame sang grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Chuyển đổi frame thành đối tượng Image
    image = Image.fromarray(frame)

    # Nhận diện hoa và dự đoán
    label = recognize_flower(image)

    # Vẽ khung xung quanh hoa và hiển thị nhãn
    if label is not None:
        cv2.rectangle(frame, top_left, bottom_right, (45, 255, 100), 2)
        cv2.putText(frame, 'Result: {}'.format(labels[label]), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255,0,0), 2)


   # Xác định vị trí hoa
    cv2.putText(frame, text, (text_x, text_y), font, font_scale, (0,0,255), font_thickness, cv2.LINE_AA)

    # Hiển thị khung ảnh kết quả
    cv2.imshow('Flower Recognition', frame)

    # Thoát khỏi vòng lặp khi nhấn phím 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    if cv2.getWindowProperty('Flower Recognition', cv2.WND_PROP_VISIBLE)<1:
        break

# Giải phóng tài nguyên và đóng cửa sổ
cap.release()
cv2.destroyAllWindows()
```

- Tạo hàm “detect_from_image()” nhận diện hoa thông qua ảnh được tải lên.

```python
def detect_from_image():
    root = tk.Tk()
    root.withdraw()

    # Chọn ảnh từ hộp thoại
    file_path = filedialog.askopenfilename()
    if file_path:
        # Đọc ảnh và chuyển đổi sang kích thước mong muốn
        image = Image.open(file_path)
        resized_image = image.resize(input_shape)

        # Chuyển đổi ảnh sang RGB
        rgb_image = resized_image.convert('RGB')

        # Nhận diện khuôn mặt và dự đoán
        label = recognize_flower(rgb_image)

        # Vẽ khung xung quanh hoa và hiển thị nhãn
        if label is not None:
            image_with_box = np.array(image)
            cv2.rectangle(image_with_box, (0, 0), (image.width, image.height), (0, 255, 0), 2)
            # Resize khung ảnh trước khi hiển thị
            display_width = 500
            display_height = 500
            image_with_box = cv2.cvtColor(image_with_box, cv2.COLOR_RGB2BGR)
            resized_display = cv2.resize(image_with_box, (display_width, display_height))
            cv2.putText(resized_display, 'Result: {}'.format(labels[label]), (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
                        0.9, (255,0,0), 2)

            # Hiển thị ảnh kết quả
            cv2.imshow('Flower Recognition', resized_display)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
```

- Tạo giao diện GUI với các nút chức năng (Button) và nhãn (Label) để thực hiện chức năng nhận diện hoa từ camera hoặc qua ảnh được tải trực tiếp.

```python
def open_camera():
    # Xử lý logic khi bật camera
    detect_from_camera()

def open_image():
    # Xử lý logic khi tải hình ảnh
    detect_from_image()

def exit_program():
    # Đợi 1 giây trước khi tắt chương trình
    window.after(1000, window.destroy)

# Tạo giao diện GUI sử dụng thư viện Tkinter
window = tk.Tk()
# Tên cửa sổ
window.title("Flower Recognition By CNN")
# Kích thước GUI
window.geometry("400x250")
# Màu nền của giao diện
window.configure(bg="#D1E9E9")  # Màu xanh nhạt
```

```python
# Label "FLOWER DETECTION USING CNN"
title_label = tk.Label(window, text="FLOWER RECOGNITION BY CNN", font=("Arial", 16, "bold"), bg="#D1E9E9", fg="#FF0000")
title_label.grid(row=0, column=1, columnspan=2, padx=10, pady=10, sticky="n")
title_label.config(anchor=tk.CENTER)

# Label "Chức năng"
function_label = tk.Label(window, text="Select :", font=("Arial", 12, "bold"), bg="#D1E9E9", padx=10, pady=5)
function_label.grid(row=1, column=1, sticky=tk.W, padx=10, pady=5)

# Label "Sử dụng Camera"
camera_label = tk.Label(window, text="1. Detect By Using Camera:", font=("Arial", 12), bg="#D1E9E9", padx=10, pady=2)
camera_label.grid(row=2, column=1, sticky=tk.W, padx=10, pady=2)

# Tạo button Camera
camera_button = tk.Button(window, text="Camera", command=open_camera, bg="#008000", fg="white", relief="solid", bd=2,
                          width=10, activebackground="#00FF00")
camera_button.grid(row=2, column=2, padx=5, pady=2)

# Label "Tải ảnh"
image_label = tk.Label(window, text="2. Detect By Using Image:", font=("Arial", 12), bg="#D1E9E9", padx=10, pady=2)
image_label.grid(row=3, column=1, sticky=tk.W, padx=10, pady=2)

# Tạo button tải hình ảnh
image_button = tk.Button(window, text="Load Image", command=open_image, bg="#0000FF", fg="white", relief="solid", bd=2,
                         width=10, activebackground="#0000CC")
image_button.grid(row=3, column=2, padx=5, pady=2)
```

```
# Tạo button "Thoát"
exit_button = tk.Button(window, text="Exit", command=exit_program, bg="#FF0000", fg="white", relief="solid", bd=2, width=10,
                        activebackground="#CC0000")
exit_button.grid(row=5, column=2, padx=5, pady=2)

# Label thông điệp
message_label = tk.Label(window, text="", font=("Arial", 12), bg="#D1E9E9", padx=10, pady=5)
message_label.grid(row=6, column=1, columnspan=2, padx=10, pady=5, sticky="n")
message_label.config(anchor=tk.CENTER)

# Khởi chạy
window.mainloop()
```

CHƯƠNG 3: TỔNG KẾT

3.1 Kết quả

Sau khi huấn luyện, mô hình đã đạt được độ chính xác trên tập kiểm tra là 83,6%. Điều này cho thấy rằng mô hình có khả năng dự đoán chính xác các loại hoa đã được huấn luyện. Và đã thiết kế được giao diện người dùng (GUI) để có thể sử dụng trực tiếp thuật toán đã huấn luyện để nhận diện hoa thì giao diện có thể nhận diện ảnh được tải lên hoàn toàn đúng và nhận diện trực tiếp thông qua camera thì kết quả vẫn tương đối khả qua vì do điều kiện môi trường xung quanh luôn thay đổi và không giống như điều kiện lúc huấn luyện mô hình.

Tổng quan về kết quả cho thấy rằng mô hình dự đoán giới tính sử dụng Convolutional Neural Network (CNN) có khả năng dự đoán chính xác và có thể được sử dụng trong các ứng dụng thực tế.

3.2 Nhận xét

Trong báo cáo này, em đã áp dụng thuật toán Convolutional Neural Network (CNN) để xây dựng mô hình thuật toán để nhận diện 10 loại hoa và xây dựng giao diện đồ họa người dùng (GUI). Em đã sử dụng bộ dữ liệu mình thu thập được từ các nguồn ảnh google và ảnh tự chụp tuy nhiên số lượng dữ liệu mà em thu thập được còn một số điểm hạn chế như độ đa dạng về loài, màu sắc, góc chụp, các điều kiện ánh sáng và số lượng chưa thực sự lớn để huấn luyện, kiểm tra và nâng cao chất lượng mô hình.

Kết quả cho thấy rằng mô hình nhận diện giới tính được huấn luyện trên đã đạt được độ chính xác khá cao trong việc nhận diện giới tính. Mô hình đã đạt được độ chính xác khoảng 83.5% trong quá trình kiểm tra trên bộ dữ liệu kiểm nghiệm. Điều này cho thấy rằng thuật toán Convolutional Neural Network (CNN) có thể

được sử dụng để nhận diện loài hoa với độ chính xác cao và có thể áp dụng trong nhiều lĩnh vực khác nhau như bảo mật, an ninh, định danh cá nhân, và nhiều lĩnh vực khác. Tuy nhiên, cần phải tiếp tục cải thiện mô hình với lượng dữ liệu cũng như xoáy sâu hơn vào các thông số của mô hình để mô hình được cải thiện thêm trước khi muốn đưa muốn đưa vào ứng dụng thực tế.

3.3 Thông tin liên hệ

Link Github báo cáo:

https://github.com/BuiChiCuongs/Detecting-Flower_AI_CNN

Link Drive báo cáo:

https://drive.google.com/drive/folders/1Wit5WBIVUId2NbUbxADys_tT8xYQKx8E?usp=sharing