TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CƠ KHÍ CHẾ TẠO MÁY

HCMUTE

BÁO CÁO CUỐI KÌ
MÔN HỌC: TRÍ TUỆ NHÂN TẠO

NHẬN DIỆN QUẢ CÀ CHUA SỬ DỤNG MẠNG NƠ RON TÍCH CHẬP CNN

GVHD: PGS.TS Nguyễn Trường Thịnh

MHP: ARIN337629_22_2_08

Họ và tên: Trần Sỹ Việt

Mssv: 20146544

Lớp: Sáng thứ 2, tiết 1 - 4

Hồ Chí Minh, tháng 5 năm 2023

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1 Giới thiệu đề tài

Báo cáo này tập trung vào việc áp dụng mạng neuron tích chập (CNN) để giải quyết bài toán phát hiện và phân loại quả cà chua từ hình ảnh. Qua nỗ lực nghiên cứu và phát triển, mô hình CNN đã được xây dựng và huấn luyện để nhận diện chính xác các loại quả cà chua từ các hình ảnh đầu vào.

Quả cà chua là một thành phần quan trọng trong ngành nông nghiệp và công nghiệp chế biến thực phẩm. Việc phân loại chính xác quả cà chua có thể giúp tăng hiệu suất sản xuất, giảm công sức lao động và đảm bảo chất lượng sản phẩm cuối cùng. Tuy nhiên, quá trình phân loại quả cà chua truyền thống yêu cầu sự can thiệp và thời gian đáng kể từ con người.

Mô hình CNN được xem là một công nghệ tiên tiến trong lĩnh vực thị giác máy tính, cho phép máy tính học tự động từ dữ liệu hình ảnh. Bằng cách sử dụng mạng CNN, mô hình của chúng tôi có khả năng phát hiện và phân loại quả cà chua với độ chính xác cao, giúp giảm thiểu sự can thiệp con người và tăng hiệu suất quá trình phân loại.

Trong báo cáo này, chúng tôi trình bày chi tiết về kiến trúc mạng CNN được sử dụng, quá trình huấn luyện mô hình, đánh giá hiệu suất và ứng dụng của mô hình trong thực tế. Ngoài ra, chúng tôi cũng đề xuất một số thách thức và hướng cải tiến tiềm năng cho mô hình để nâng cao hiệu suất và độ tin cậy trong tương lai.

Với tiềm năng của mô hình phát hiện quả cà chua bằng CNN, hy vọng rằng công nghệ này sẽ đóng góp vào việc tự động hóa quá trình phân loại quả cà chua và mang lại những lợi ích đáng kể trong lĩnh vực nông nghiệp và công nghiệp chế biến thực phẩm.

1.2 Mục đích nghiên cứu

- Xây dựng mô hình phát hiện quả cà chua bằng mạng nơ ron tích chập CNN
- Phát triển kỹ năng ứng dụng trí tuệ nhân tạo hỗ trợ phát triển tối ưu hóa sản xuất trong nghành nông nghiệp
- Xây dựng ứng dụng web app để dễ dàng thao tác và sử dụng.

1.3 Phương pháp nghiên cứu

- Tìm hiểu về trí tuệ nhân tạo và mạng nơ ron tích chập CNN
- Tìm hiểu các môi trường huấn luyện như Google Colab
- Tìm hiểu về cách xây dựng một Web app.
- Xây dựng mô hình phân loại và đưa ra dự đoán kết quả.

1.4 Thông tin đề tài

- Dữ liệu:

https://drive.google.com/drive/folders/1RUtGq6frerGcZ2jBeCY-Ir4rCHap-LVm?usp=share_link

- Github:

https://github.com/viettran02/AI_project/blob/main/Tomato_Detection.ipynb

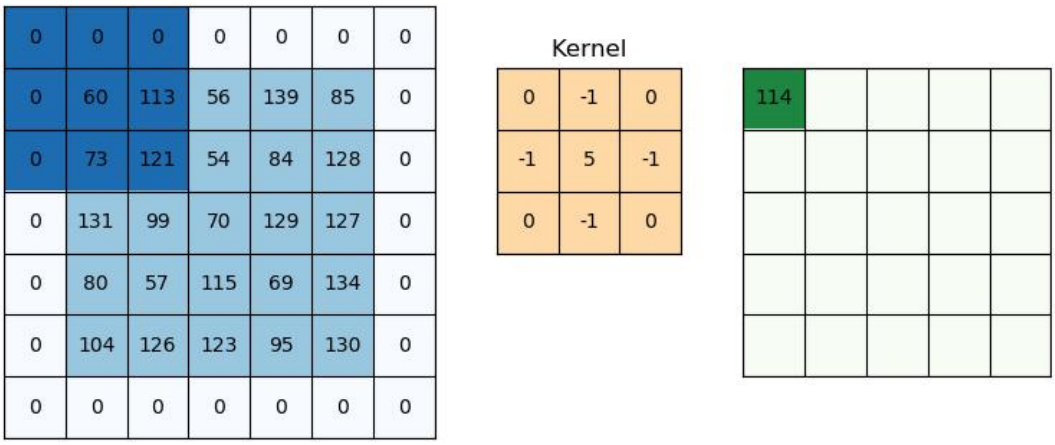
-

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Thuận toán CNN - Convolutional Neural Network

2.1.1 Khái niệm

Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh. Để tìm hiểu tại sao thuật toán này được sử dụng rộng rãi cho việc nhận dạng (detection), chúng ta hãy cùng tìm hiểu về thuật toán này.



Hình 1: Ma trận Convolutional

Đây là một "cửa sổ" sử dụng trượt trên ma trận nhằm lấy được những thông tin chính xác và cần thiết nhất mà không phải chọn đặc trưng (feature). Convolution hay nhân tích chập là cách mà những lớp Convolutional này nhân những phần tử trong ma trận. Sliding Window hay kernel là dạng ma trận có kích thước nhỏ, sử dụng trong nhân tích chập với ma trận hình ảnh.

2.1.2 Feature trong CNN

Feature là đặc trưng, mạng CNN sẽ so sánh dựa vào từng mảnh và các mảnh như vậy được gọi là feature. Thay vì phải tiến hành khớp các bức ảnh lại với

nhau thì mạng CNN sẽ xác định được sự tương đồng thông qua tìm kiếm thô những đặc trưng khớp với nhau bằng hai hình ảnh tốt hơn. Một feature là mộ hình ảnh dạng mini (những mảng 2 chiều nhỏ). Những feature này đều tương ứng với một khía cạnh nào đó của hình ảnh và chúng có thể khớp lại được với nhau.

2.1.3 Các lớp cơ bản của mạng CNN.

Convolutional layera:

Lớp này là phần quan trọng nhất của toàn mạng CNN, nó có nhiệm vụ thực thi các tính toán. Các yếu tố quan trọng trong lớp Convolutional là: padding, stride, feature map và filter map.

Mạng CNN sử dụng filter để áp dụng vào các vùng của ma trận hình ảnh. Các filter map là các ma trận 3 chiều, bên trong đó là những tham số và chúng được gọi là parameters. Stride tức là bạn dịch chuyển filter map theo từng pixel dựa vào các giá trị từ trái qua phải.

Padding: Thường, giá trị viền xung quanh của ma trận hình ảnh sẽ được gán các giá trị 0 để có thể tiến hành nhân tích chập mà không làm giảm kích thước ma trận ảnh ban đầu.

Feature map: Biểu diễn kết quả sau mỗi lần feature map quét qua ma trận ảnh đầu vào. Sau mỗi lần quét thì lớp Convolutional sẽ tiến hành tính toán.

Relu Layer:

Lớp ReLU này là hàm kích hoạt trong mạng CNN, được gọi là activation function. Nó có tác dụng mô phỏng những nơ ron có tỷ lệ truyền xung qua axon. Các hàm activation khác như Leaky, Sigmoid, Leaky, Maxout,.. tuy nhiên hiện nay, hàm ReLU được sử dụng phổ biến và thông dụng nhất.

Hàm này được sử dụng cho những yêu cầu huấn luyện mạng nơ ron với những ưu điểm nổi bật điển hình là hỗ trợ tính toán nhanh hơn. Trong quá trình dùng hàm ReLU, bạn cần chú ý đến việc tùy chỉnh những learning rate và dead unit. Những lớp ReLU được dùng sau khi filter map được tính và áp dụng ReLU lên các giá trị của filter map.

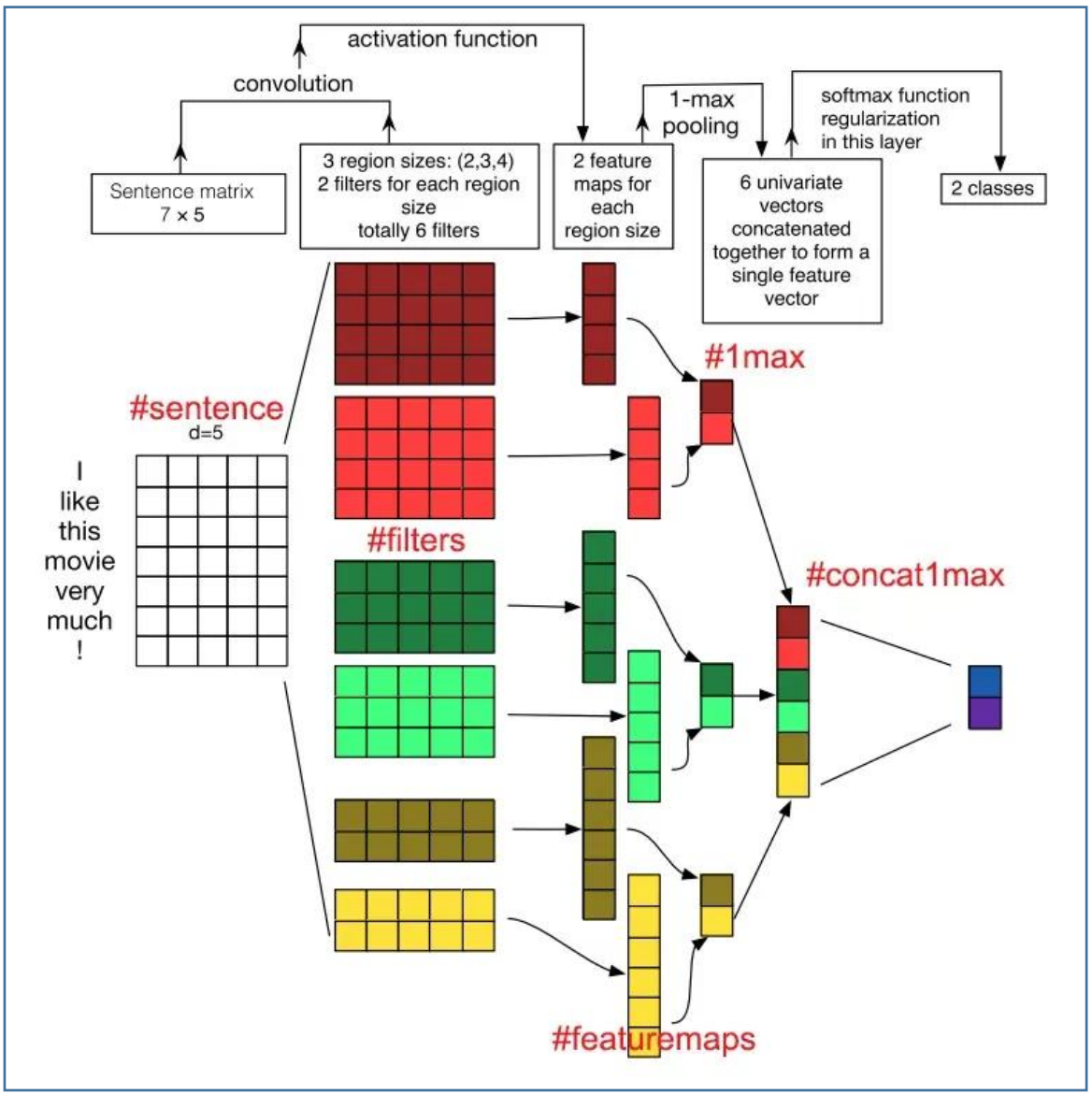
Pooling layer:

Khi ma trận ảnh đầu vào có kích thước quá lớn, các lớp Pooling layer sẽ được

đặt vào giữa những lớp Convolutional để làm giảm những parameters. Hiện, hai loại lớp Pooling được sử dụng phổ biến là Max pooling và Average.

Fully connected layer:

Đây là lớp có nhiệm vụ đưa ra kết quả sau khi hai lớp Convolutional và Pooling đả nhận được ảnh truyền. Khi này, ta sẽ thu được một model đọc được thông tin của ảnh. Để có thể liên kế chúng cũng như cho nhiều đầu ra hơn ta sẽ sử dụng Fully connected layer.



Hình 2: Các lớp cơ bản của mạng CNN

2.1.4 Cấu trúc của mạng CNN

Mạng CNN là tập hợp những Convolutional layer xếp chồng lên nhau, đồng thời mạng sử dụng những hàm như ReLU và Tanh để kích hoạt các trọng số

trong các node. Các lớp này sau khi qua các hàm activation sẽ có trọng số trong những node và có thể tạo ra những thông tin trừu tượng hơn đến với các lớp kế tiếp trong mạng.

Mạng CNN có tính kết hợp cà tính bất biến. Tức là, nếu cùng một đối tượng mà sử dụng chiếu theo các góc độ khác nhau thì sẽ có ảnh hưởng đến độ chính xác. Với dịch chuyển, co giãn hay quay ma trận ảnh thì lớp Pooling sẽ được dùng để hỗ trợ làm bất biến các tính chất này. Chính vì vậy mà mạng CNN sẽ đưa ra những kết quả có độ chính xác tương ứng với từng mô hình.

Trong đó, lớp Pooling sẽ có khả năng tạo tính bất biến với phép dịch chuyển, co giãn và quay. Còn tính kết hợp cục bộ sẽ cho thấy những cấp độ biểu diễn, dữ liệu từ thấp đến cao với mức trừu tượng thông qua Convolution từ filter. Mạng CNN có những lớp liên kết nhau dựa vào cơ chế Convolution.

Các lớp tiếp theo sẽ là kết quả từ những lớp trước đó, vì vậy mà bạn sẽ có những liên kết cục bộ phù hợp nhất. Trong quá trình huấn luyện mạng, CNN sẽ tự học hỏi những giá trị thông qua filter layer dựa theo cách thức mà bạn thực hiện.

Cấu trúc cơ bản của một mô hình mạng CNN thường bao gồm 3 phần chính bao gồm:

- Trường cục bộ/ Local receptive field: Lớp này sử dụng để tách lọc dữ liệu, thông tin hình ảnh để từ đó có thể lựa chọn các vùng có giá trị sử dụng hiệu quả cao nhất.
- Trọng số chia sẻ/ Shared weights and bias : Lớp này hỗ trợ làm giảm các tham số đến mức tối thiểu trong mạng CNN. Trong từng lớp convolution sẽ chứa các feature map riêng và từng feature thì sẽ có khả năng phát hiện một vài feature trong hình ảnh.
- Lớp tổng hợp/ Pooling layer: Đây là lớp cuối cùng và sử dụng để làm đơn giản các thông tin output. Tức là, sau khi tính toán xong và quét qua các layer trong mạng thì pooling layer sẽ được dùng để lược bỏ các thông tin không hữu ích. Từ đó cho ra kết quả theo kỳ vọng người dùng.

2.2 Thư viện Tensorflow và Keras

2.2.1 Thư viện Tensorflow

TensorFlow là một thư viện mã nguồn mở phổ biến trong lĩnh vực học máy và trí tuệ nhân tạo. Được phát triển bởi Google Brain, TensorFlow cung cấp một nền tảng mạnh mẽ để xây dựng và triển khai các mô hình học máy phức tạp.

Với TensorFlow, người dùng có thể dễ dàng xây dựng và huấn luyện các mạng neuron sâu, bao gồm cả mạng neuron tích chập (CNN) được sử dụng trong phân loại hình ảnh. Thư viện cung cấp các lớp, hàm và công cụ tiện ích cho việc xây dựng mạng neuron, quản lý dữ liệu, tối ưu hóa và đánh giá mô hình.

TensorFlow hỗ trợ nhiều ngôn ngữ lập trình như Python, C++, Java và Go, cho phép người dùng lựa chọn ngôn ngữ ưa thích của họ để phát triển mô hình. Ngoài ra, TensorFlow cũng tích hợp nhiều công cụ và thư viện bổ sung để giúp phân tích dữ liệu, trực quan hóa kết quả và triển khai mô hình trên nền tảng khác nhau

Với sự phát triển và sự hỗ trợ của cộng đồng mạnh mẽ, TensorFlow đã trở thành một công cụ quan trọng trong lĩnh vực học máy và trí tuệ nhân tạo. Việc sử dụng TensorFlow giúp giảm thời gian và công sức trong việc xây dựng và triển khai mô hình, đồng thời mang lại hiệu suất và độ chính xác cao cho các ứng dụng học máy

2.2.2 Thư viện Keras

Keras là một thư viện mã nguồn mở, dễ sử dụng và mạnh mẽ cho việc xây dựng mạng neuron và mô hình học sâu. Ban đầu được phát triển như một dự án độc lập, Keras đã được tích hợp vào TensorFlow từ phiên bản TensorFlow 2.0 trở đi.

Keras cung cấp một giao diện cao cấp (high-level API) giúp người dùng dễ dàng xây dựng, huấn luyện và đánh giá các mô hình học máy. Với Keras, bạn có thể xây dựng mạng neuron theo cách modul và linh hoạt, từ việc xác định kiến trúc mô hình, thêm các lớp mạng, kết nối các lớp, cho đến việc cấu hình quá trình huấn luyện và tối ưu hóa mô hình.

Keras hỗ trợ các loại mô hình học sâu phổ biến như mạng neuron tích chập (CNN), mạng neuron hồi quy (RNN) và mạng nơ-ron tái cấu trúc (Recurrent Neural Network - RNN). Ngoài ra, Keras cũng cung cấp nhiều lớp và hàm tiện ích để xử lý dữ liệu, tiền xử lý và trực quan hóa kết quả.

Sự kết hợp giữa Keras và TensorFlow mang lại những lợi ích vượt trội, như tích hợp mạnh mẽ giữa các tính năng của cả hai thư viện, khả năng tận dụng tối đa các tính năng của TensorFlow trong quá trình huấn luyện và triển khai mô hình.

Với đơn giản, tính linh hoạt và hiệu suất cao, Keras là một lựa chọn phổ biến cho các nhà nghiên cứu và lập trình viên trong việc phát triển mô hình học máy và mô hình học sâu.

2.3 Thư viện Flask

Flask là một framework web phát triển bằng ngôn ngữ Python, giúp xây dựng ứng dụng web dễ dàng và nhanh chóng. Flask được thiết kế đơn giản, nhẹ nhàng và linh hoạt, cho phép người dùng xây dựng các ứng dụng web từ những ứng dụng nhỏ đến những hệ thống phức tạp.

Flask hỗ trợ việc xử lý các yêu cầu HTTP, tạo các tuyến đường (routes) và điều hướng các yêu cầu đến các hàm xử lý tương ứng. Nó cũng cung cấp các công cụ và thư viện hỗ trợ để làm việc với các yêu cầu và phản hồi HTTP, quản lý phiên làm việc, xử lý biểu mẫu, và nhiều tính năng khác.

Với Flask, bạn có thể tạo các ứng dụng web tương tác với người dùng, nhận dữ liệu từ các biểu mẫu và xử lý chúng, lưu trữ và truy xuất dữ liệu từ cơ sở dữ liệu, và trả về các trang HTML hoặc dữ liệu JSON tới người dùng.

Flask cũng hỗ trợ việc mở rộng ứng dụng thông qua các plug-in và tiện ích bổ sung. Nó cung cấp một cách tiếp cận tổ chức mô-đun, cho phép bạn xây dựng ứng dụng dễ dàng chia thành các thành phần riêng biệt và mở rộng linh hoạt theo nhu cầu.

Với sự linh hoạt, đơn giản và cộng đồng phát triển mạnh mẽ, Flask là một lựa chọn phổ biến trong việc xây dựng ứng dụng web với Python, từ các dự án nhỏ cá nhân cho đến các dự án lớn và thương mại.

CHƯƠNG 3: XÂY DỰNG ĐỀ TÀI

3.1 Xây dựng mô hình trên GG Colab

Mô hình được huấn luyện trên Google Colab

3.1.1 Dữ liệu.

Dữ liệu của mô hình được em thu thập trực tiếp trên Google bao gồm các hình ảnh các trạng thái khác nhau của quả cà chua về màu sắc, kích thước, giống loài và hình ảnh các loại trái cây khác được đưa vào mô hình huấn luyện. Do hình ảnh thu thập không đồng đều nên em đã scale về kích thước 150x150. Các dữ liệu đưa chia ra các thư mục train, test, và validation như bảng bên dưới.

3.1.2 Huấn luyện mô hình

Bước 1: Khai báo các thư viện cần thiết.

```
import keras
import os
import numpy as np
import matplotlib.pyplot as plt
from glob import glob
from keras.models import Sequential
from keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img
```

```
from keras.layers import Dense, Dropout, Flatten,Activation
from keras.layers import Conv2D, MaxPooling2D, Conv3D,
BatchNormalization
from tensorflow.keras.preprocessing.image import
ImageDataGenerator, load_img,img_to_array
```

Bước 2: Khai báo đường dẫn để lấy dữ liệu được upload lên GG Driver

```
test_path="/content/drive/MyDrive/Data_Tomato/Test"
train_path="/content/drive/MyDrive/Data_Tomato/Train"
val_path="/content/drive/MyDrive/Data_Tomato/Val"
```

Bước 3: Tăng cường dữ liệu

Tăng cường dữ liệu là cách tạo ra nhiều ảnh hơn từ ảnh gốc. Mục đích của tăng cường dữ liệu là nhằm tăng độ chính xác hơn cho mô hình huấn luyện.

```
train_datagen = ImageDataGenerator(rescale=1/255,
 rotation_range=0.2, #xoay hình
 horizontal_flip=True,
 vertical_flip=True,
```

```
 zoom_range=0.2)
test_datagen = ImageDataGenerator(rescale=1/255)
```

Các thao tác được thực hiện để tăng cường dữ liệu:

- *Lệnh rescale*: scale ảnh về 1/255
- *Lệnh 'rotation_range':* Phép xoay hình với góc xoay tối đa là 0.2 radian. Điều này giúp tăng sự đa dạng của dữ liệu huấn luyện và giảm hiện tượng quá khớp.
- *Lệnh horizontal_flip=True*: Phép lật ngang hình ảnh theo trục hoành. Điều này tạo ra các phiên bản lật ngang của hình ảnh, giúp tăng tính đa dạng và khả năng tổng quát hóa của mô hình.
- *Lệnh vertical_flip=True:* Phép lật dọc hình ảnh theo trục tung. Tương tự như lật ngang, phép lật dọc giúp tăng tính đa dạng và khả năng tổng quát hóa của mô hình.
- *Lệnh zoom_range=0.2:* Phép zoom hình ảnh với tỷ lệ tối đa là 0.2. Điều này tạo ra các phiên bản thu nhỏ hoặc phóng to của hình ảnh, giúp mô hình học được khả năng nhận diện và tổng quát hóa tốt hơn.

Sau đó chúng ta tiến hành lưu dữ liệu train_set, test_set, val_set với các dữ liệu đã được tăng cường.

```
train_set = train_datagen.flow_from_directory(train_path,
 target_size=(150,150),
 batch_size=10,
 class_mode='categorical')
test_set = test_datagen.flow_from_directory(test_path,
 target_size=(150,150),
 batch_size=10,
 class_mode='categorical')
val_set = train_datagen.flow_from_directory(val_path,
 target_size=(150,150),
 batch_size=10,
 shuffle=True,
 class_mode='categorical')
```

```
Found 1153 images belonging to 2 classes.
Found 130 images belonging to 2 classes.
Found 21 images belonging to 2 classes.
```

Kết quả: Train_set có 1153 ảnh thuộc 2 lớp, tương tự test_set có 130 ảnh, và val_test có 21 ảnh.

Bước 4: Xây dựng mô hình CNN

```
model = Sequential()
model.add(Conv2D(64, kernel_size=(3,3), activation='relu',
input_shape=(
150,150, 3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(16, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(8, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Activation("relu"))
model.add(Dense(1024))
model.add(Dropout(0.4))
model.add(Dense(2))
model.add(Activation("softmax"))
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

- *model = Sequential():* Tạo một đối tượng mô hình Sequential.
- *model.add(Conv2D(64, kernel_size=(3,3), activation='relu', input_shape=(150,150, 3))):* Thêm một lớp Conv2D với 64 bộ lọc (filters), kích thước kernel là (3,3), hàm kích hoạt là ReLU. Đây là lớp đầu tiên của mô hình và có input_shape là (150,150,3), chỉ định kích thước của ảnh đầu vào.
- *model.add(MaxPooling2D(pool_size=(2,2))):* Thêm một lớp MaxPooling2D để giảm kích thước không gian của đầu ra bằng cách lấy giá trị lớn nhất từ mỗi khối pooling có kích thước (2,2). Quá trình này giúp giảm số lượng tham số và giảm chiều dữ liệu. Các bước 2 và 3 lặp lại thêm ba lần với các số lượng bộ lọc và kích thước kernel khác nhau để trích xuất đặc trưng từ ảnh.

- *model.add(Flatten()):* Thêm một lớp Flatten để biến đổi đầu ra của các lớp trước thành một vectơ 1 chiều, chuẩn bị cho các lớp Fully Connected tiếp theo.

- *model.add(Activation("relu")):* Thêm một lớp Activation với hàm kích hoạt ReLU để tăng tính phi tuyến trong mô hình.
- *model.add(Dense(1024)):* Thêm một lớp Dense với 1024 nơ-ron, là một lớp Fully Connected, có chức năng học và trích xuất thông tin từ đặc trưng đã được biểu diễn trước đó.
- *model.add(Dropout(0.4)):* Thêm một lớp Dropout với tỷ lệ 0.4 để ngẫu nhiên tắt 40% các nơ-ron trong quá trình huấn luyện, nhằm tránh hiện tượng quá khớp (overfitting).
- *model.add(Dense(2)):* Thêm một lớp Dense với 2 nơ-ron, là lớp đầu ra cuối cùng. Trong trường hợp này, mô hình được huấn luyện để

Các thông số cụ thể được thể hiện trong bảng bên dưới.

Bước 5: Huấn luyện mô hình

Mô hình huấn luyện với số lần học là 100 và số bước cho mỗi lần học là 50

```
history = model.fit_generator(train_set,
 steps_per_epoch=50,
 epochs=100,
 validation_data=test_set,
 validation_steps=200//10)
```

```
Epoch 1/100
<ipython-input-24-f089c45b9860>:1: UserWarning: `Model.fit_generator` is deprecated and will be
  history = model.fit_generator(train_set,
50/50 [==============================] - ETA: 0s - loss: 0.6196 - accuracy: 0.6880WARNING:tensor
50/50 [==============================] - 11s 163ms/step - loss: 0.6196 - accuracy: 0.6880 - val
Epoch 2/100
50/50 [==============================] - 5s 92ms/step - loss: 0.5399 - accuracy: 0.7465
Epoch 3/100
50/50 [==============================] - 4s 77ms/step - loss: 0.5216 - accuracy: 0.7820
Epoch 4/100
50/50 [==============================] - 5s 91ms/step - loss: 0.4811 - accuracy: 0.7860
```

.....

```
Epoch 94/100
50/50 [==============================] - 5s 97ms/step - loss: 0.2293 - accuracy: 0.9020
Epoch 95/100
50/50 [==============================] - 4s 72ms/step - loss: 0.2016 - accuracy: 0.9160
Epoch 96/100
50/50 [==============================] - 4s 73ms/step - loss: 0.1390 - accuracy: 0.9500
Epoch 97/100
50/50 [==============================] - 5s 90ms/step - loss: 0.1753 - accuracy: 0.9229
Epoch 98/100
50/50 [==============================] - 4s 73ms/step - loss: 0.1664 - accuracy: 0.9400
Epoch 99/100
50/50 [==============================] - 4s 72ms/step - loss: 0.1734 - accuracy: 0.9270
Epoch 100/100
50/50 [==============================] - 4s 78ms/step - loss: 0.2206 - accuracy: 0.9067
```

3.2 Xây dựng webapp với thư viện Flask.

Xây dựng một ứng dụng web sử dụng Flask là một cách phổ biến để triển khai các mô hình học máy và học sâu. Với Flask, chúng ta có thể tạo một giao diện người dùng đơn giản để tải lên ảnh và đưa ra dự đoán từ mô hình.

Bước 1: Cài đặt thư viện Tlask.

- Mở termial và gõ dòng lệnh: *pip install tflask*

Bước 2: Tạo file app.py và viết chương trình.

```
from flask import Flask, render_template, request
from keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
app = Flask(__name__)

dic = {0 : 'Cà Chua', 1 : 'Không phải quả cà chua'}
model = load_model('Tomato_Detection.h5')
model.make_predict_function()
def predict_label(img_path):
  i = image.load_img(img_path, target_size=(150,150))
  i = image.img_to_array(i)/255.0
  i = i.reshape(1, 150,150,3)
  p = model.predict(i)
  predicted_class = np.argmax(p, axis=1)
  return dic[predicted_class[0]]
# routes
@app.route("/", methods=['GET', 'POST'])
def main():
  return render_template("index.html")
@app.route("/about")
def about_page():
  return "Please subscribe  Artificial Intelligence Hub..!!!"
@app.route("/submit", methods = ['GET', 'POST'])
def get_output():
  if request.method == 'POST':
    img = request.files['my_image']
    img_path = "static/" + img.filename
    img.save(img_path)
    p = predict_label(img_path)
  return render_template("index.html", prediction = p,
img_path = img_path)
if __name__ =='__main__':
  #app.debug = True
  app.run(debug = True)
```

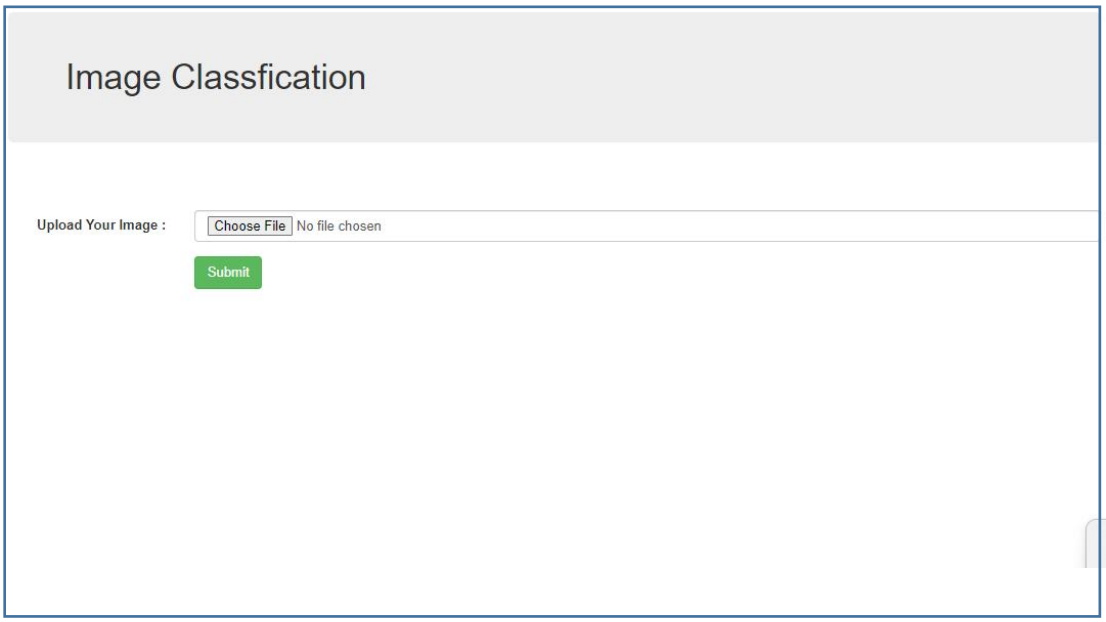
Bước 3: Tạo template HTML làm giao diện để giao tiếp với web app

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Image Classification</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/boot
strap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.
min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootst
rap.min.js"></script>
</head>
<body>

<div class="container">
  <h1 class="jumbotron bg-primary">Image Classfication</h1>
  <br><br>
  <form class="form-horizontal" action="/submit" method="post"
enctype="multipart/form-data">
    <div class="form-group">
      <label class="control-label col-sm-2" for="pwd">Upload
Your Image :</label>
      <div class="col-sm-10">
        <input type="file" class="form-control"
placeholder="Hours Studied"  name="my_image" id="pwd">
      </div>
    </div>
    <div class="form-group">
      <div class="col-sm-offset-2 col-sm-10">
        <button type="submit" class="btn btn-
success">Submit</button>
      </div>
    </div>
  </form>
  {% if prediction %}
  <img src="{{img_path}}" height="400px" width="400px">
  <h2> Your Prediction   : <i> {{prediction}} </i></h2>
  {% endif %}
</div>
</body>
</html>
```

Bước 4: Chạy chương trình để lấy đường dẫn truy cập vào web app.

- Mở terminal và gõ dòng lệnh: *python app.py*
- Sao chép đường dẫn ***http://127.0.0.1:5000*** và gán vào trình duyệt



Hình 3: Kết quả sao khi truy cập vào link web app

CHƯƠNG 4: KẾT QUẢ

4.1 Kết quả dự đoán trên google colab.

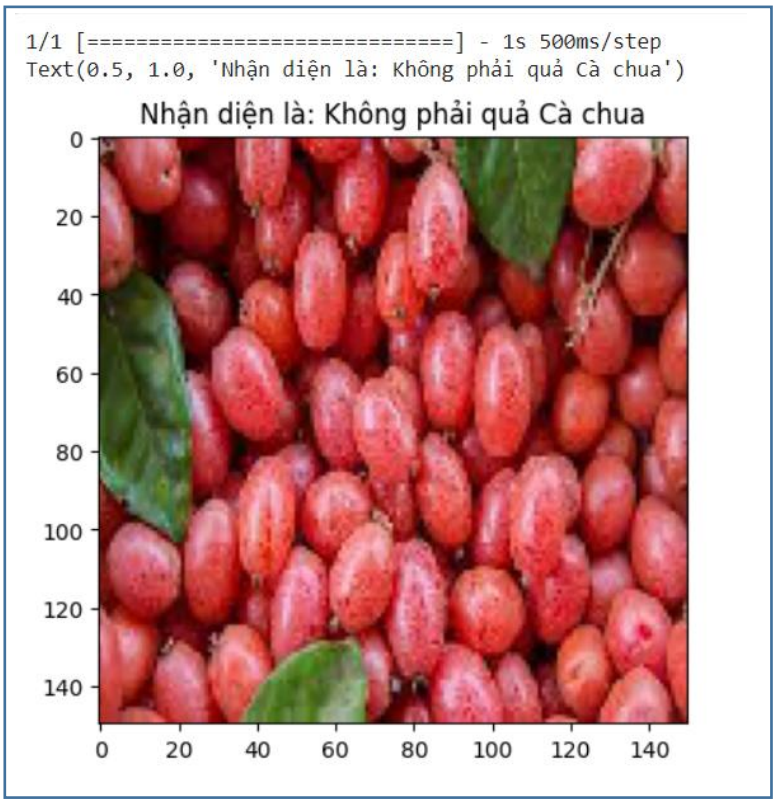
Mô hình dự đoán đạt trên quả trên 90% và có kết quả tương đối chính xác.

Mô hình dự đoán đươc lưu vào: Tomato_Detection.h5

Chương trình lấy dữ liệu từ file h5 và đưa ra dự đoán:

```
map_dict = { 0: "Cà Chua",1: "Không phải quả Cà chua"}
img =
load_img('/content/drive/MyDrive/quanhot.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
model100=load_model("/content/drive/MyDrive/Tomato_Detection.h
5")
prediction = model100.predict(img).argmax()
plt.title("Nhận diện là: {}".format(map_dict[prediction]))
```

Kết quả

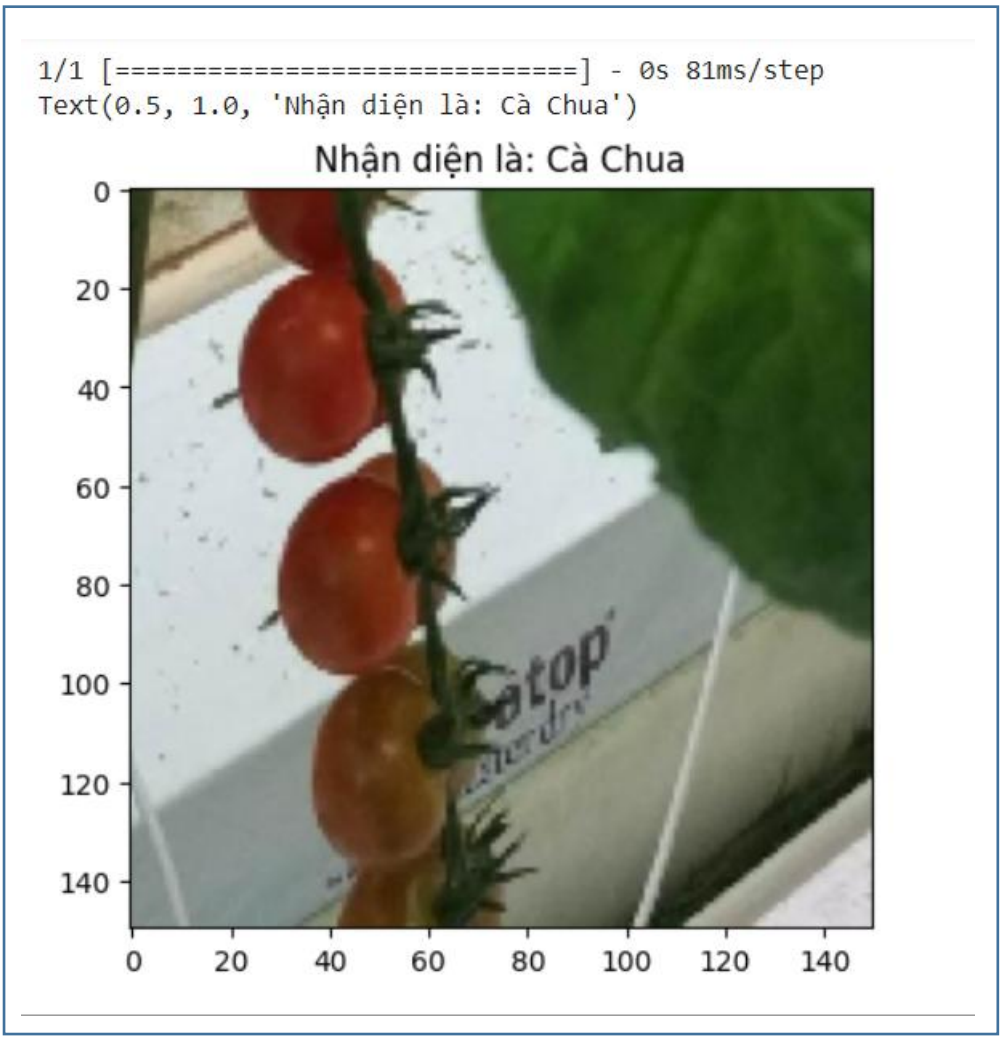


Hình 4: Hình ảnh dự đoán với đầu vào là quả nhót.

Chương trình lấy dữ liệu từ file h5 và đưa ra dự đoán:

```
map_dict = { 0: "Cà Chua",1: "Không phải quả Cà chua"}
img =
load_img('/content/drive/MyDrive/tomato804.png',target_size=(1
50,150))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255
model100=load_model("/content/drive/MyDrive/Tomato_Detection.h
5")
prediction = model100.predict(img).argmax()
plt.title("Nhận diện là: {}".format(map_dict[prediction]))
```

Kết quả:

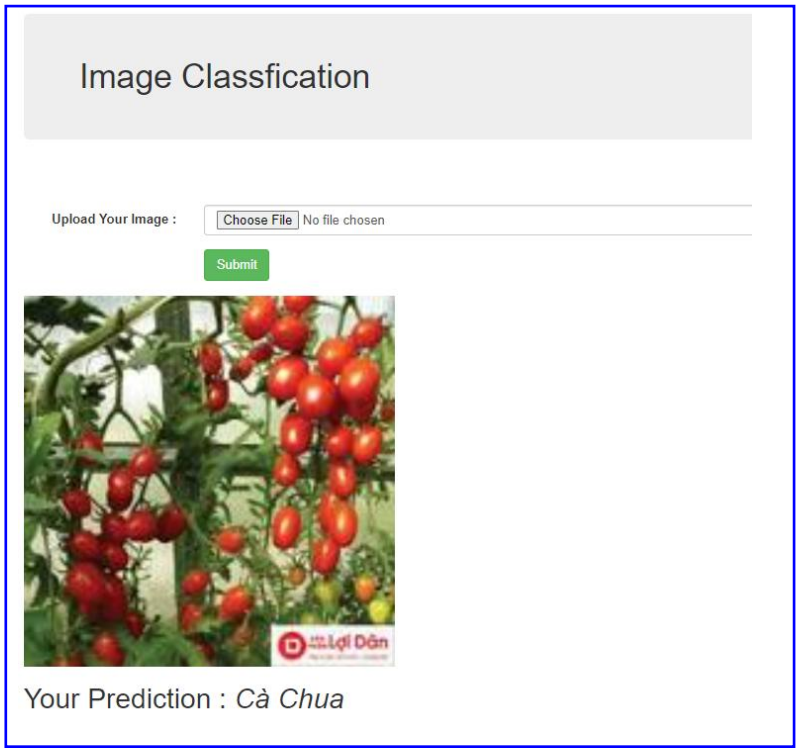


Hình 5:Kết quả dự đoán với ảnh đầu vào là quả cà chua

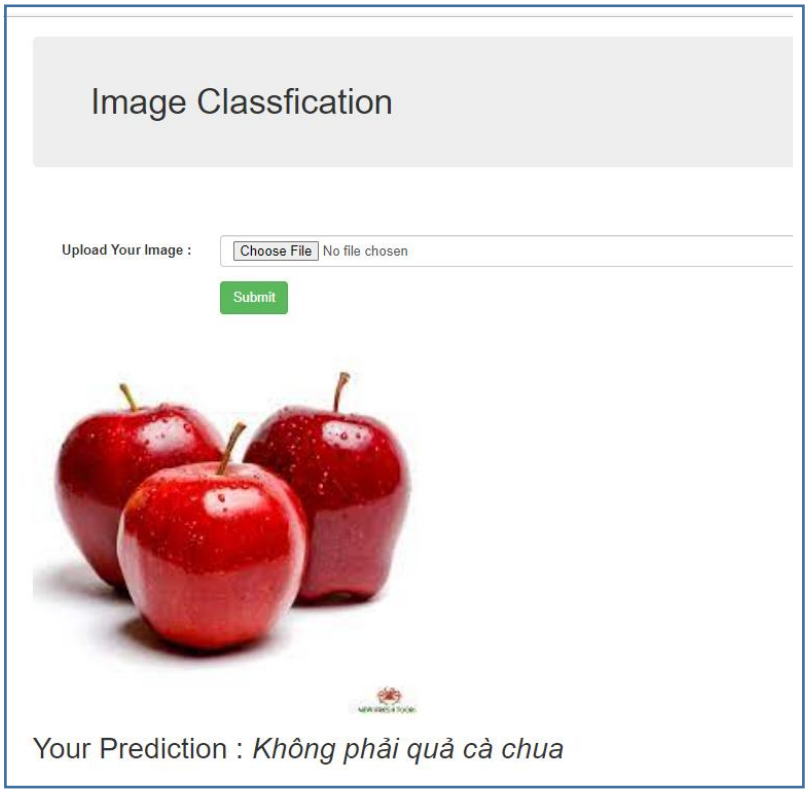
4.2 Kết quả dự đoán trên web app.

- Upload file ảnh từ máy tính và chạy thử:

- Kết quả:



Hình 6: Kết quả dự đoán với ảnh đầu vào là ảnh cà chua



Hình 7: Kết quả dự đoán với ảnh đầu vào là quả táo

Tài Liệu Tham Khảo

1. Thuật toán CNN là gì? Thông tin cấu trúc của mạng CNN (fpt.edu.vn)
2. Image Classification using CNN Keras | Full implementation (youtube.com)
3. Image Classification Webapp | Flask (youtube.com)
4. Deploy Machine Learning Models using Flask (youtube.com)