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

    Mounted at /content/drive
```

```
!unzip "/content/drive/MyDrive/Nutrition Image Analysis using CNN and Rapid API-20221103T055609Z-001.zip"
```

```
inflating: Nutrition Image Analysis using CNN and Rapid API/Nutrition Analysis Using Image Classification/Training/D
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/APPLES/n07740461_10074.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/226_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/236_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/23_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/228_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/227_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/234_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/ORANGE/n07749192_4808.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/225_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/221_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/233_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/245_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/231_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/232_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/220_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/22_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/223_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/239_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/214_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/224_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/229_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/217_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/212_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/219_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/222_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/216_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/218_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/215_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/20_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/ORANGE/n07749192_479.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/200_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/198_100.jpg
```

```
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/1_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/208_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/21_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Nutrition Analysis Using Image Classification/Training/D
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/195_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/204_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/73_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/19_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/190_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/192_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/184_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/179_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/199_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/194_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/181_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/187_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/193_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/197_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/177_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/18_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/180_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Nutrition Analysis Using Image Classification/Training/D
inflating: Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/WATERMELON/17_100.jpg
inflating: Nutrition Image Analysis using CNN and Rapid API/Nutrition Analysis Using Image Classification/Training/D
```

```
pip install keras
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-packages (2.9.0)
```

Import The ImageDataGenerator Library

```
from keras.preprocessing.image import ImageDataGenerator
```

Configure ImageDataGenerator Class

```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

Apply Image DataGenerator Functionality To Trainset And Testset

```
x_train=train_datagen.flow_from_directory(r'/content/Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET',tar
x_test=train_datagen.flow_from_directory(r'/content/Nutrition Image Analysis using CNN and Rapid API/Dataset/TEST_SET',targe
```

```
     Found 4118 images belonging to 5 classes.
     Found 929 images belonging to 5 classes.
```

```
print(x_train.class_indices)
```

```
     {'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}
```

```
print(x_test.class_indices)
```

```
     {'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}
```

```
from collections import Counter as c
c(x_train .labels)
```

```
     Counter({0: 995, 1: 1354, 2: 1019, 3: 275, 4: 475})
```

Importing the Libraries:

```
import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout
from keras.preprocessing.image import ImageDataGenerator
```

Initializing The Model

```
model = Sequential()
```

Adding CNN Layers

```
classifier = Sequential()
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Conv2D(32, (3, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Flatten())
```

Adding Dense Layers

```
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=5, activation='softmax'))
```

```
classifier.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0         
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dense_1 (Dense)             (None, 5)                 645

=================================================================
Total params: 813,733
Trainable params: 813,733
Non-trainable params: 0
_________________________________________________________________
```

Configure The Learning Process

```
classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Train The Model

```
classifier.fit_generator(
    generator=x_train, steps_per_epoch = len(x_train),
    epochs=20, validation_data=x_test, validation_steps = len(x_test)
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generator` is deprecated and wi
  This is separate from the ipykernel package so we can avoid doing imports until
Epoch 1/20
129/129 [==============================] - 21s 154ms/step - loss: 0.6914 - accuracy: 0.7241 - val_loss: 0.5389 - val_ac
Epoch 2/20
129/129 [==============================] - 20s 151ms/step - loss: 0.4479 - accuracy: 0.8339 - val_loss: 0.5202 - val_ac
Epoch 3/20
129/129 [==============================] - 18s 140ms/step - loss: 0.3922 - accuracy: 0.8494 - val_loss: 0.4651 - val_ac
Epoch 4/20
129/129 [==============================] - 19s 143ms/step - loss: 0.3577 - accuracy: 0.8616 - val_loss: 0.4221 - val_ac
Epoch 5/20
129/129 [==============================] - 20s 152ms/step - loss: 0.3248 - accuracy: 0.8747 - val_loss: 0.4287 - val_ac
Epoch 6/20
```

```
129/129 [==============================] - 18s 141ms/step - loss: 0.3082 - accuracy: 0.8834 - val_loss: 0.4126 - val_ac
Epoch 7/20
129/129 [==============================] - 18s 141ms/step - loss: 0.2825 - accuracy: 0.9002 - val_loss: 0.3460 - val_ac
Epoch 8/20
129/129 [==============================] - 20s 151ms/step - loss: 0.2803 - accuracy: 0.8951 - val_loss: 0.4413 - val_ac
Epoch 9/20
129/129 [==============================] - 18s 141ms/step - loss: 0.2541 - accuracy: 0.9034 - val_loss: 0.4130 - val_ac
Epoch 10/20
129/129 [==============================] - 20s 152ms/step - loss: 0.2585 - accuracy: 0.9009 - val_loss: 0.3649 - val_ac
Epoch 11/20
129/129 [==============================] - 19s 151ms/step - loss: 0.2467 - accuracy: 0.9031 - val_loss: 0.3626 - val_ac
Epoch 12/20
129/129 [==============================] - 20s 152ms/step - loss: 0.2289 - accuracy: 0.9106 - val_loss: 0.3666 - val_ac
Epoch 13/20
129/129 [==============================] - 20s 152ms/step - loss: 0.2065 - accuracy: 0.9245 - val_loss: 0.4046 - val_ac
Epoch 14/20
129/129 [==============================] - 18s 141ms/step - loss: 0.2096 - accuracy: 0.9240 - val_loss: 0.3558 - val_ac
Epoch 15/20
129/129 [==============================] - 20s 152ms/step - loss: 0.1942 - accuracy: 0.9240 - val_loss: 0.3349 - val_ac
Epoch 16/20
129/129 [==============================] - 18s 143ms/step - loss: 0.1791 - accuracy: 0.9315 - val_loss: 0.3631 - val_ac
Epoch 17/20
129/129 [==============================] - 18s 141ms/step - loss: 0.1814 - accuracy: 0.9271 - val_loss: 0.3848 - val_ac
Epoch 18/20
129/129 [==============================] - 20s 152ms/step - loss: 0.1593 - accuracy: 0.9420 - val_loss: 0.3418 - val_ac
Epoch 19/20
129/129 [==============================] - 18s 143ms/step - loss: 0.1662 - accuracy: 0.9376 - val_loss: 0.3457 - val_ac
Epoch 20/20
129/129 [==============================] - 18s 142ms/step - loss: 0.1574 - accuracy: 0.9376 - val_loss: 0.3037 - val_ac
<keras.callbacks.History at 0x7fc0bc8bf290>
```

Save The Model

```
classifier.save('nutrition.h5')
```

Test The Model

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model = load_model("nutrition.h5")
```

```
img = image.load_img("/content/Nutrition Image Analysis using CNN and Rapid API/Dataset/TEST_SET/ORANGE/n07749192_1081.jpg",
grayscale=False,target_size=(64,64))

x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)
pred = np.argmax(model.predict(x), axis=-1)
pred
```

```
    1/1 [==============================] - 0s 89ms/step
    array([2])
```

```
index=['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON']
result=str(index[pred[0]])
result
```

```
    'ORANGE'
```

Importing Libraries

```
from flask import Flask, render_template, request
import os
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import requests
```

Creating our flask application and loading our model by using the load_model method

```
app = Flask(__name__, template_folder="templates")
model = load_model('/content/nutrition.h5')
```

```
print("Loaded model from disk")
```

```
Loaded model from disk
```

Routing To The Html Page

```
@app.route('/')
def home():
  return render_template('/content/drive/MyDrive/Nutrition Analysis Using Image Classification/Flask/templates/home.html')

@app.route('/image1', methods=['GET', 'POST'])
def image1():
  return render_template("/content/drive/MyDrive/Nutrition Analysis Using Image Classification/Flask/templates/image.html")


@app.route('/predict', methods=['GET', 'POST'])
def launch():
  if request.method=='POST':
    f = request.files['file']
    basepath = os.path.dirname('__file__')
    filepath = os.path.join(basepath, "uploads", f.filename)
    f.save(filepath)

    img = image.load_img(filepath, target_size=(64,64))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    pred = np.argmax(model.predict(x), axis=1)
    print("prediction", pred)
    index = ['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON']

    result = str(index[pred[0]])

    x = result
    print(x)
    result = nutrition(result)
    print(result)
    return render_template("/content/drive/MyDrive/Nutrition Analysis Using Image Classification/Flask/templates/0.html", sh

def nutrition(index):

  url = "https://calorieninjas.p.rapidapi.com/v1/nutrition"

  querystring = {"query":index}

  headers = {
    "X-RapidAPI-Key": "521c31f652msh1c7495bea1f7905p109714jsn0aa70be1becb",
    "X-RapidAPI-Host": "calorieninjas.p.rapidapi.com"
    }

  response = requests.request("GET", url, headers=headers, params=querystring)

  print(response.text)
  return response.json()['items']


if __name__ == "__main__":
  app.run(debug=False)
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
INFO:werkzeug: * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Colab paid products - Cancel contracts here

Executing (1m 35s) Cell > run() > run_simple() > inner() > serve_forever() > serve_forever() > select()