```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "7c4b4d5c",
   "metadata": {},
   "outputs": [],
   "source": [
    "from keras.preprocessing.image import ImageDataGenerator"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 65,
   "id": "6ad3fa30",
   "metadata": {},
   "outputs": [],
   "source": [
    "train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)\n",
    "test_datagen =ImageDataGenerator(rescale=1./255)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 66,
   "id": "ed4d4293",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Found 4118 images belonging to 5 classes.\n",
      "Found 929 images belonging to 5 classes.\n"
     ]
    }
   ],
   "source": [
    "x_train = train_datagen.flow_from_directory(\n",
    "    r'C:\\Users\\HP\\Documents\\DataSet\\TRAIN_SET-20221021T095226Z-001\\TRAIN_SET',target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')\n",
    "x_test = test_datagen.flow_from_directory(\n",
    "    r'C:\\Users\\HP\\Documents\\DataSet\\TEST_SET-20221021T095157Z-001\\TEST_SET',target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 67,
   "id": "a70e2030",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "{'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}\n"
     ]
    }
   ],
   "source": [
    "print (x_train.class_indices)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 68,
   "id": "e78e7713",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "{'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}\n"
     ]
    }
   ],
   "source": [
    "print(x_test.class_indices)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 69,
   "id": "a1c2cb32",
   "metadata": {
    "scrolled": true
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "Counter({0: 995, 1: 1354, 2: 1019, 3: 275, 4: 475})"
      ]
     },
     "execution_count": 69,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "from collections import Counter as c\n",
    "c(x_train.labels)\n",
    "\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 70,
   "id": "dc6912bd",
   "metadata": {},
   "outputs": [],
   "source": [
    "from keras.models import Sequential\n",
    "from keras.layers import Dense\n",
    "from keras.layers import Conv2D\n",
    "from keras.layers import MaxPooling2D\n",
    "from keras.layers import Flatten\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 71,
   "id": "36f1d672",
   "metadata": {
    "scrolled": false
   },
   "outputs": [],
   "source": [
    "import numpy as np"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 72,
   "id": "f1021b6b",
   "metadata": {},
   "outputs": [],
   "source": [
    "import tensorflow"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 73,
   "id": "d809b19d",
   "metadata": {
    "scrolled": true
   },
   "outputs": [],
   "source": [
    "from tensorflow.keras.models import Sequential"
   ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 74,
 "id": "7239e628",
 "metadata": {
  "scrolled": true
 },
 "outputs": [],
 "source": [
  "from tensorflow.keras import layers"
 ]
},
{
 "cell_type": "code",
 "execution_count": 75,
 "id": "eba820ec",
 "metadata": {},
 "outputs": [],
 "source": [
  "from tensorflow.keras.layers import Dense, Flatten"
 ]
},
{
 "cell_type": "code",
 "execution_count": 76,
 "id": "73bd47f6",
 "metadata": {},
 "outputs": [],
 "source": [
  "from tensorflow.keras.layers import Conv2D,MaxPooling2D,Dropout"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 77,
  "id": "7798ac53",
  "metadata": {},
  "outputs": [],
  "source": [
   "from keras.preprocessing.image import ImageDataGenerator"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 78,
  "id": "6769dbb1",
  "metadata": {},
  "outputs": [],
  "source": [
   "model=Sequential()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 79,
  "id": "455498d3",
  "metadata": {},
  "outputs": [],
  "source": [
   "model.add(Conv2D(32,3,3,input_shape=(64,64,3),activation='relu'))"
  ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 80,
 "id": "fae88e4b",
 "metadata": {},
 "outputs": [],
 "source": [
  "model.add(MaxPooling2D(pool_size=(2,2)))"
 ]
},
{
 "cell_type": "code",
 "execution_count": 81,
 "id": "2d090a28",
 "metadata": {},
 "outputs": [],
 "source": [
  "model.add(Flatten())"
 ]
},
{
 "cell_type": "code",
 "execution_count": 82,
 "id": "354f4509",
 "metadata": {},
 "outputs": [],
 "source": [
  "#model.add(Dense(output_dim=128,activation='relu',init='random_uniform'))"
 ]
},
{
 "cell_type": "code",
 "execution_count": 83,
 "id": "874c0651",
 "metadata": {},
 "outputs": [],
 "source": [
  "#model.add(Dense(output_dim=1,activation='sigmoid',init='random_uniform'))"
 ]
},
{
 "cell_type": "code",
 "execution_count": 84,
 "id": "8704fcd5",
 "metadata": {},
 "outputs": [],
 "source": [
  "classifier = Sequential()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 85,
 "id": "a926bbb5",
 "metadata": {},
 "outputs": [],
 "source": [
  "classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))\n",
  "classifier.add(MaxPooling2D(pool_size=(2, 2)))"
 ]
},
{
 "cell_type": "code",
 "execution_count": 86,
 "id": "eb0b8bad",
 "metadata": {},
 "outputs": [],
 "source": [
  "classifier.add(Conv2D(32, (3, 3), activation='relu'))\n",
  "\n",
  " \n",
  "\n",
  "\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": 87,
 "id": "b1ca8e74",
 "metadata": {},
 "outputs": [],
 "source": [
  "classifier.add(MaxPooling2D(pool_size=(2, 2)))\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": 88,
 "id": "b84fa096",
 "metadata": {},
 "outputs": [],
 "source": [
    "classifier.add(Flatten())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 90,
   "id": "9acf57b9",
   "metadata": {},
   "outputs": [],
   "source": [
    "classifier.add(Dense(units=128, activation='relu'))\n",
    "classifier.add(Dense(units=5, activation='softmax'))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 91,
   "id": "23c1f8c5",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Model: \"sequential_4\"\n",
      "_________________________________________________________________\n",
      " Layer (type)                Output Shape              Param #   \n",
      "=================================================================\n",
      " conv2d_6 (Conv2D)           (None, 62, 62, 32)        896       \n",
      "                                                                 \n",
      " max_pooling2d_8 (MaxPooling  (None, 31, 31, 32)       0         \n",
    " 2D)                                                            \n",
    "                                                                \n",
    " conv2d_7 (Conv2D)           (None, 29, 29, 32)        9248      \n",
    "                                                                \n",
    " max_pooling2d_9 (MaxPooling  (None, 14, 14, 32)       0         \n",
    " 2D)                                                            \n",
    "                                                                \n",
    " flatten_6 (Flatten)         (None, 6272)              0         \n",
    "                                                                \n",
    " dense_4 (Dense)             (None, 128)               802944    \n",
    "                                                                \n",
    " dense_5 (Dense)             (None, 5)                 645       \n",
    "                                                                \n",
    "=================================================================\n",
    "Total params: 813,733\n",
    "Trainable params: 813,733\n",
    "Non-trainable params: 0\n",
    "_________________________________________________________________\n"
   ]
  }
 ],
 "source": [
  "classifier.summary()\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": 92,
 "id": "b5aeb89e",
 "metadata": {},
 "outputs": [],
   "source": [
    "classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 93,
   "id": "cdefcc8a",
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "C:\\Users\\HP\\AppData\\Local\\Temp\\ipykernel_13540\\1881611701.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.\n",
      "  classifier.fit_generator(\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Epoch 1/10\n",
      "824/824 [==============================] - 41s 48ms/step - loss: 0.6049 - accuracy: 0.7739 - val_loss: 0.5004 - val_accuracy: 0.8116\n",
      "Epoch 2/10\n",
      "824/824 [==============================] - 41s 50ms/step - loss: 0.4198 - accuracy: 0.8473 - val_loss: 0.4984 - val_accuracy: 0.8235\n",
      "Epoch 3/10\n",
     "824/824 [==============================] - 43s 52ms/step - loss: 0.3806 - accuracy: 0.8555 - val_loss: 0.4393 - val_accuracy: 0.8332\n",
     "Epoch 4/10\n",
     "824/824 [==============================] - 38s 45ms/step - loss: 0.3472 - accuracy: 0.8681 - val_loss: 0.4119 - val_accuracy: 0.8504\n",
     "Epoch 5/10\n",
     "824/824 [==============================] - 37s 45ms/step - loss: 0.3323 - accuracy: 0.8779 - val_loss: 0.4166 - val_accuracy: 0.8428\n",
     "Epoch 6/10\n",
     "824/824 [==============================] - 37s 44ms/step - loss: 0.3072 - accuracy: 0.8834 - val_loss: 0.4251 - val_accuracy: 0.8428\n",
     "Epoch 7/10\n",
     "824/824 [==============================] - 39s 47ms/step - loss: 0.2990 - accuracy: 0.8905 - val_loss: 0.5612 - val_accuracy: 0.7783\n",
     "Epoch 8/10\n",
     "824/824 [==============================] - 48s 58ms/step - loss: 0.2626 - accuracy: 0.8975 - val_loss: 0.3379 - val_accuracy: 0.8859\n",
     "Epoch 9/10\n",
     "824/824 [==============================] - 49s 59ms/step - loss: 0.2587 - accuracy: 0.9004 - val_loss: 0.4058 - val_accuracy: 0.8568\n",
     "Epoch 10/10\n",
     "824/824 [==============================] - 46s 56ms/step - loss: 0.2350 - accuracy: 0.9053 - val_loss: 0.3717 - val_accuracy: 0.8794\n"
    ]
   },
   {
    "data": {
     "text/plain": [
      "<keras.callbacks.History at 0x1dea14127c0>"
     ]
    },
    "execution_count": 93,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "classifier.fit_generator(\n",
   "        generator=x_train,steps_per_epoch = len(x_train),\n",
   "        epochs=10, validation_data=x_test,validation_steps = len(x_test))"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 94,
  "id": "27dfd665",
  "metadata": {},
  "outputs": [],
  "source": [
   "classifier.save('nutrition.h5')"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 95,
  "id": "3dbf69b2",
  "metadata": {},
  "outputs": [],
  "source": [
   "from tensorflow.keras.models import load_model"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 96,
  "id": "86cd3d3d",
  "metadata": {},
  "outputs": [],
  "source": [
   "from keras.preprocessing import image\n",
   "from tensorflow.keras.preprocessing import image \n",
   "model = load_model(\"nutrition.h5\")"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 97,
  "id": "5da73d02",
  "metadata": {},
  "outputs": [
   {
    "data": {
     "image/png":
```

"iVBORw0KGgoAAAANSUhEUgAAAEAAAABACAIAAAAlC+aJAAAZFklEQVR4nO166a9l2XXXWmtPZ7jn3
vfue/Vq6upyd3vCA4mTGMdyIseOiB0TWwoGxwiEIQODZEXiQxQkIv6GxPlg4BNCQpGMBAhMFISI5RiM
jYd0t6vtavfkrqpX9aY7n3EPa/Hhvq70YLrbQciK5KWje6V7z9fVb0157o4o4jAX2ShHzaA/1f5kQI/bPmR
Aj9s+ZECryUi8v81U/+FZ0C//lsHCAQCAAigAwESALHA4MGWAAApBquVtAmN4dST1bKp2zs3n5gfucSk
h8ZGbCGwuv7gO65deTO4DFiGJDqjIGAjkAGRBJgQ7OtEhT8Av1EAGZABpUONwASM0WuF2K+Go6Nn
bzxxeni41+NqsbQK6uViaGpzEOADu9E1WSwSFsWw6troF7lqyjxNaz889BM/+ab3vQ+MW5PLwBghjAD
m9YJ6bQXu3yCAhMwSWHodisNvfWN297mwPq5XZ+nJ4wpoPIifL59fnxzsTLHrj44PTZWNH3b898rhYj
ONB81CsbrtPVfryxfjQ0989lHMyzSZmulBMtYclL/wD/4xZNWS9AQkhAAA1r4GFT+ACwkH9vWwPP3Gl7
54sMG+XTqd/OpoR/Mq86fHsyS0nN29+pfe+uzN71a5m7z16j+sKU/ThrMlQQaJMYeaS0c88/9ryZOG
sy34SSh7ptrqbhG5/9l/zAlXf/yifajvM8R8R7TXRPUDMPDMl/6Hv3dbZsdydtQTL5azRNE4Ndmr7hS3M8CS
kWLsv34E3rvc6Qemqwv2xtGdtowffv+P6/L4OB7vvNW5B86v5Q+Em9s+XB9XF73zx27hWFZbq4enxYklZ
eeXa9RsPv/FTn/oU0WvnmJcrlCL4wg9Ot2l/9PvV/ftIeqhfhIl+Wx0/jHMN6X5mmEA8Yu9K6/XS3vH01/5ErZftkK4Q
yvm9xXxbPvDMOGvHWKczsHVQrFgqUZDzWZmbjQ/+Bj/Oy/c8SdiZwJ+eSN4aTMu76pHL4j5Gwhw8aEDuKiVcHFvbFSU
2+K3MgBoMnGwSDfAEhewh3KdoMUDAeSAJBAzRhenBQfnM1N2wYNlyEbbbGzSWHHz8oY+E0kh1BZ6Kei5CDMEydH8qsQT
S+nQP6lzvkgkdahugfSBbT82xFQQSOK0q0ek/tRMFVqDX1g0DbwCJeHM6MWPx7hPQsMoYECl0RqQ9X6wLWyuNOfivcT
Y3sKO6jTW8vRUedq6YpYGZ8gP2SE5Wl2AUBSvgbhJGv+3OndKvEk06RmE+Do0EQl9n91k1NwzxtaLt84pRgPXxeN2
1zkS/IeXg6Ky0hpqQ3c5BJcupV3UY30pwngD5/aTyydR7CmKFO2i0AyAOjYvmRDZTsqBVIlAvW4n/nLsPaPJ5YS3oTFK
+gmaeVYP/VAHQFDl+FNfswzPx16CP2Ep/n/3D5tpuT4ggHFVqHVpw8bNQZWrz8O0yRq24j6ZWzzPyN2E/
oLmIzZVGGG6xmqNbh9ocFNhIqq0+9r0onm+JYoDZvI8OnNW97nBrk/18H/cFwI/wWb2wwBt9/lcDROzjDGtvO6u
bTVxBx8Tnxf2E5ctmDRrJg3pxjv4A5dV70KyRonOI0hVOyOt4cbV0pIvFEHaR3ZVRKpz9uirEvtJzAaMTdzqDcA
WJe97HHvXv2MVqHeX7oMTOEgSiOqmyFNLUCmz71cjHHV7TL+YNuhxMOE8UOPq/+/yHIuCtaaIcYbwA8YK
Ubg9PX1HoEL+Np4HhSPf6nmnydZI6Pn5utapxyXF+Trt7GrF9GWavpqOR/IiyNH3v0RVxN7H+nrPspeRvoXtbO+
5UXAT4zngwhQo0wAAAC0uvc3sgvZaDQkbvtXSlbdP7

```
P/etr6wfe0Keecav65OnjqAcCtT8hmDe8QbcrQ2Ts3oITYprDRWq3NfVZKecYlaovG07Beb5W7fuHt1L8y
Yjt6N/erI/a5/ZlYXdzUZaivai8Ih1mRmTCaDqPKQeI5cjGzPCDKK35Yja9aD6E7eK94avHJ3d/rlf/1XmEor8
1RSQRGYYjh6/YQ7vpbP13edupflmvmglH8lD+QMHxQbvtWE9nw1WkUODIt167ngKXcAjOHvqjpvJZuin
Ru/p/fzgwaQnd/7LrZ0LB5c7Mp6GVRMGLiNOdZWqdtOeKSTf9W3bQh9DCIjIzFESCCtFKcW6reM6o6fjz
S/8z3d8+K+9DPArGBD+2uc+98Bi/diffA0WQyaAEXPhidaXP/jOQz4J6h7rzagY+bYZWqZmHlqB/nTxnUV
63L9t/Jbb3z5m5o33psh6a3ppCndluNl33SyFGqIBYJbAyAYME3YpRAJBINYAEGPUWrMEVAQKldWocT
WfXQzh6ItfboXf9Vd/iYhCCFmWwSubucf+6D9ls5PbX/lTXCQXlY1Y2sxq2p1WxbWyGOfCLsSxBmckc2Bz
tnnK4sn6keqqXfLzT9x0WkU/GIUQw3o2azb1+uRZv/pe7BbsO6A66TbqLuouMTADAKEQCgmQAAEqQJ
UpMoRGobFKJDkgjsP66PD4698MIYjIFv33YQAWd9XsbPbU7dSVmdWAUTtMfpjuT3jsuxAmo2s+Ot8dE
RJCtJBpAc188uTRA3tXvO7VPDptez8AyyQvE13RctLVp0pNiUZtOEuovWSCtMJISIgEACgQiQBAEEXrHCQ
p9AbBag9sEBoewhB27x1//vOf//jHP+6933YZOoLXybYKCojf+/JXN48/3T76bB6NWD8KZtDp6u6FtpiE9z
7SFruGqjwCoinzSxu+x7iSWobGZyJ6BGo89U+fhfUGEoDTGlWzGTx+zzkXaYyQUNYK0GMCIhIZsRIBIIzC2
pg8ggdGo6OClgOhIiGK0WqdAhA6o83J4fHou//ryT/O3vKBjwUADUEDICCAxP7uYfPUs5M6nT1/dKCLQ
GAujceXdj2lbOL0qMpl1DtjGAg0RJiUuc6G1enZ5YNyGObZZX98OnvjT7559af3FmeLet3s7u5IwEJIkRqEE
RABPQAKKCICFCIAaPp+78L+2WI+diMETkQIAEIABEB+iMRAhhgRnTZKp5P61o3vvO0DH0MBQdICJCgS/
Om3b+Lde0//70dHTIqoUzy9tjdU1hoDWtV1v487K2pE2GLmpdekLYDTIgO7fMdFP0F9Nls0y6VzztvQdR
0zI0Pb9cYYYWFhUqgBfEwsaLVJIFU58v1ASSCxNirGyAhaaRHgKGhUSoKYUKsUotW6O1o27hRCQzoPSQ
iABKG0Zv7003zn7li5qhxFgitvfpgsuqoYTXeLojTJxkXUVJGqNI6KogAxvlfVaKqUYiW6LFU1yfauXr9+ve/76
XQaYwSApBCsjgTbS8k5AwZJEWkkJQAxTarx+V+yLbW0JYEZOAGHyCFKiDEE2yTctF/6z/8eIIQkZNingLDe
HHCjut40HgxOH7norV8ZqspqPXSdkm6xAEYTbLJlEo5sDDmrrS4KrMaIUJugObng19KOL+4Q0WyxWqcE
MbXrTewHp7QlFQlEkYgYY6xDYzEhsLUD0kASUjRKO0YVECNKQhKdEvRejM4lsREZudEwXw/HJ5A4RCZI
bBTc+OpX7jz93LNPfBeVyoti/+LBznS6d+nAk+R7O9n+TrG/uzpbZugMGEJNaBA0ghqGQSkERahIay0icG
Hsruz1kK5cuVJoK4hlVdks64Zhtdkk5iT84u4lz/Ou6+q6FgRBYJAkzIRbaZpGC+Z5vlouU0opJSJM0fu6fv5b
j5eFoyElYOD1vEigByatbOaOT0+0NcvNev/yxS6FlsNafLfuC3GODaBFsIiGSDvnUkpAKIiXH3ygDUOXkdoZ
YWZXqxX4OAQfObk8U0Zfu/5gAmEQRkgv9GR1XVtr8zxnAAaIzEmEFbJCRHzXO/9yu940y3WZF5gYEyM
JxMBt9+y3byZJhFnG61Xhm7KPFbvxzsSW+f6li6vNeres1qdza61xzhZ5JkVzuMRGgDJFpcKC0DCzUghaA
eHx6UmW5xnpbrURH53SmTZbozZdm4RPzk4jShBu+w4IQwhd1523MCL3GRCELnjPSUSeffKp3dE4t45
D3Ja6Vb0orFJNo/1AmGiIYXF6zE3dr2sHJgp3Xdd1XQIhnzAkiElS4pgcmdhETJhYQNQ2zSmllKaiKHo/dM
MgCFOVURdmh/dQwFq7d3AhgSSQKByFt3DJ6Lprh2EQkbIsY4wiwgiMoJ3dv3iQXugQQgiKCAEQAFiAJa
XAMWFIfb0hQBqpfHl2zFKiGQcXfNeHzA6sJmoypKAQlU++6QQBQQB4aFoTmPoeYkgMEo3qTTCVLnby
qgw5BN/Fuq2wdGJ76brlRiPFlDzKoCABRkAGFUWBLiO5dd0QAHCCiBDRt+HsaAYASdhzAqP6GAQgxBQ
iKlUmKgQsRlI9KwENCPVmZfygNarCTKqdZExZFETUp4GZQwiBQfdDygcJTnpFHQXkkDynoDhEiUZAADN
ramCaVlkT5k/NcQgsQdAkOLeuIIiAIBTVSIuKPglCYmYBQSB8yVrxnCsiABFEIorMzMy4/eJhGECEAGBsTe
oaykntZpHA1+2w3Ggk0kpENKATdBHYDyqGtFrhchVmMz9fStspSQhxdfduqNfr+cwoFYCboR+Nqy2ChK
K0VtZETsqaRMCE67ap2yZIYhBRJJpEUUrp/upqf38/hKCUEhEi2s4nq6par9dElFJi5u3NWkCIU5XnPbIeZd
QbB5h6P5/Ps0ujxWrV9z0VjnKny3zYbKCyfp146JVVjJIkxU1dksqcafshIsu8Tcu6X25IICD4FMkanyIZHWK
UF6hgABBhZla4pUWTuj9HOTs701prrSXEbYgz83w+Lyfj/qUUaQbkzpvI2hifWzvO69nChjgS4mUSieNx1
Qxt0SdN6M8WLjimOE+N02acFX5V+75PDKuNtwJmiEXQt49m5JvACcal1K2PLEiJgQUTWi+CmtpuGCkC
AGQRBBAAgAQiCAJglXaooA+gKWqMQVhEIcV+cJQpDQnBhwFi0ggghD7GduiVUrOT05HNUHjT1EVW9
n2Pud6dTJi5XqyggL5JlMOIjG/62WJjSfVtF9o+xkjaQB+bdejbzkS2WbEc/HmWvG91AABgZqXOPZ5xCx6
2U2wBSAiahUCAhSPDC35FRHTecgMAKK2BUBMAa9JVPh6Pw7ppY9q9uLtZLEUMxFQWxWq5splJQ+9
TvVcenC1nTpW8rhVi8n5g7vt+FDEG5r7jnsMixiYS0uAHNyrrtEkAjMDbF8u5AlbrlAAABODcrxJva5kAm
MQIKJGBQARBIwBsncrTuUW0s0BIKTJrGiTNFvOh60kghqCdLSZVVY72xju5sShglFYhLk9Oy8z5vhMNQ+
gJpFksHpjuze/eGeqNCrA5Xs1uHRUm9yAJqWsDi2xtuGVg683bzwQSQRhkW56FELVCRYzgOfWS2JBH8
XAer9sHBWA83SknY+NsAtEoUE13jjhcODgYjud5Na7rOqaUUsKMlkufj/N2U6Omup85GAfjaxvG1w762
aD7QINfHd7zvhPGzXp1abR/uLgniKJ0SAgiTC9ELYLA+VCHiAiJE7w4phEBz5WEjpgRI0DUEAky+TMXYpD
ZYlFNsoPJGwWB2NCl69e9Am/McoiqKkCRJbWblS34hntylNhnudbFBa3KTAwu23C2yXLT8AYReRmMsp
qp1OVqtmEAbTNgJCJWGmWkJMOodVQmIqKSIWlRQUBAMSgWiowxQVKokhjPJopDBQAB2FprgTwnj
```

CxDIMBawagq0IWVTeItEQCW1fTytVXXFJPSFfnO3hS1WtUbpRQzp5Scc6PRaFQUQ9126x5aXs5niQeVq
ZSnUAQ9Kc10PASf6i6J8NYl8Dza7htvK+Px+JUFa+shzHw/UrcSY4wxKiIiQkWiiQiQdDDuIx//hLFECkAE87
2L+WTixa/rzdHx8dlsFjmdnZ0Nw3Dnzh3v/cnJSYzeahdqrzzpQKkNWZZRrtax1plzVamclcQ2cy7LgGhI0W
Rum9rv153zCvr9RETUCyIvyHaXYFu50OhBktHKS8omF8BtxyqJwYymDz7spjvVdNzHYMscrXZlMZ1Oy7I
cjUbj8ZiZizxDz92sUZ0a1YXduGEWDGZ5NonrZraY273xOg1gFBiFRoUUN039Mga+L/SXqcF8P9+eP6iRU
FEA7pFFgii9f/2RekghBoLEoKi4cFCORi634/F4u1zKsmxrkhDCer0uy7Jpmr7vq9FoebbUXqNXfhP6OsQ+F
aiBZdO1SeF9rNrZi5cubXHgC/J6FHjxuBZfJCzCIt57bcx7//rftG6kNZFYxgge7Wo83hhXY7CZRcRls2l9ODo
83cn32lrqXimQwKEJw9W3PLLc3PYm7VUX9JoCm7OhLcToRVdKFit91C+GFB26+ni1fXe6LwhCyAgMEo
URUSFpQZWEELergqBAKYWIIhJjZGaFJIqUNSry/ni6nGSeomWJyJQAAUEZmr7henb5mgdKqPbGO75uu
q6bTCZaa+dc3/d115MyIckwBC1Gizk5PEWPELDcvxStBW1IK+I0ypxTlBuNnIhoC+XFpj3XylAncUD2JJ7k/
sI/0rkjMXNVVUop0ecRHK3qqvwX/9YnjduOeIUSaABJGC+8/ceG/as7b3xTb21AqSZj55zWervaEJH10aJft
tTL4vCsOWvuPncv1LFbdDqZsIqxSUianBokmNyyRVaMRrZwy7JExKqqROR+15lAhDAKJxBQRAIoIIkh8bb
X2BqOiAYFQjhI0hcmd6y5/BPvGhhACIFJAAE5AvdAD//UTw9l7i5ekMJSptu2zfP81q1bq9Xq5OQENh7W
wwgsNmF/cqFddaGNoU2zozkeN7gYlIAHxtLgyGLpokMcue0Qc7lcbr3ixe6uGTSDBdpeJoFNUKDO8c8G
nttwEk1IhEbXHH7n93+vNRZpu2fNOgMAUgUoIIA9K5cPpK9xVQy97Bxkx6dnFtWVnb2HyuLJJ580zizaR
eCwWnaXdndHRUkjPLx1O7GIQUomtTxbDqORCSDa2kAJh64s8yBsi3Lde+dc3TZklADoIACwncQBgCG1
pUUlcUCJISiOVgUNqJzYeNzP/sbf/y1QdnRuANRQvDyvvf0Dv+gnU3UwrfaygzdcS4bsqDg6Oz2anSqjyeh
+GK4/9NBoNDo7Ozs6Orp161aMcbXcKDJnZ/O27d/znvcYY7apzBijMkvOjHcmfdsRwND3VhsUoJdWNy
LaFr7tlRC2kwvWJIosDd7qd/3SR9/y8x9+GeBXJGZjr7/nffHCdC7D84sTPS7VKN+5sNe0rRDazJHRp4uZUk
prvXXTS5cuKWWY5dLFK/t7Bzdu3AghaK3zPFdK1WFohn5Vb5g5+qAAtVLIAiyIuA3xbfaMIEOKQVgUJYV
RIyska1RmuXSbzH3k07/Z08u38F8+XmdIo4ce7pplM/R49zTL8+7OiZJkigxCiiDlpErAwLzNLU1d900bg8Q
A986OuRlYIyIy4nw+19aW00mz7pTWCoQTuKKs22Ybryx83t+LMDMZgy/0raCICZggAiPwWVX+6j/7594
5IxFQv5oCBNwBXn37O91mve4e1XnEVZuhysui2zRA0rV1OR4t5sssy5pNXZblerEs81FRFPW6HY3Gq24
NACJQlGXTdT4G6yxqpCCSoK43SqkACV5Us85TLKEIigggECIgCEFWFmT173z2X0g16QEsJ1CvqgCAmQA
MPu2/94PJFKtnb+apDmcr6bnXoKMoyBgzldkIsHvlonNu0NDWSWYrGkKv0/jCdLNaW1Kx6StjfYjMXgA2
fZvneT5y/RCil1E13mw2qBU50w+DzQofAQ0YIoIYEZOEYNVyd+e3f+8zUuUAkgOCci+D+389apBSihxc19
z54z/iw7vN0fHieJNHaFbrYjQKNHjv+74XEe89gVM+yRD2Lh7cunO7KArFkGJ02rQpMjMiDsNQlmVYd4O
PomzvgzJaRHTmej+YzHUcDZLWihEk9P7B/Xsj909//1+ByrU7ryevxPlqZyVaACXikr/xh/8RVqfNrSO3GuJi
025qcLwdeGz7AxDrV3WmTdN1YFTTNKn3050dFGjaLsuyuq6NMSEE8JwEy9EOIK2bWlsTOAEiEBpOEO
TC9WvP9Ct8cPobv/sZyVwPpF/18M2rHvZgFiAgQA7PPvrl+luPhedPaNOTj4B+NptlWRZjTCl1vaS6c6gGjl
hmRqm+7UZ5sZwvCu2qqlosFqvV6urVq7du3dHGdUMCJGW0siamJIRAyBRiwq5yD37k5z726U9bzEBAI
Al2BKM/lwIvlaHuZze+efz41yf9+vj5QwRlXHZ8OptMJsubz1+5cmW1WqHAraO7+5cvbrpWCdggEqIxTp
Fp294Y161mQdGKJCBaMYIsGrVVUbivsp/9tb/z9p99f57vvGJD/s/HwEtlA8kMbabxv/2bf13duzXJi7PD2
3mKvm4WlPMQTAIJsQ1dtTNZ1OvdoqpPZmRN2/ZKmbbbpjbFd3TiTYQRg7BWitRtF2YW9t/zYOz/4W/9
o0MqDBcHqdTTeP7ACvD0wtw0nYFkvcbP+d5/53YNJtVydctPvZ6Pl4XGeu7pvQavUDTmoVWy1tsGnpu
n7vp/Y8SwOK0O9pnpofv6jH/vl3/wnyZgByYJEQAGwDOp1H0b8gc7MASAACiCvUTkAC4CJ+9PTbJLLrTv
/4Q8+d+Obj44z3YZBZw5DCuum3N/tu0EEg5cY42I2+8Q//LWf+eQv+8wQam1sBxSBEsAOA4AACwiAeb
0a/AAKhHMSAoC3ITAIg9ImSyIdZgjQe++sVaEZOOauTBJK0IzQNF1ZViCICDPoCnDQ+gyt5BS6lCkFwsAQ
cyFgAQ7AGeSvDQgAAP4PAf+XyWlar8YAAAAASUVORK5CYII=\n",
      "text/plain": [
       "<PIL.Image.Image image mode=RGB size=64x64 at 0x1DE9F9C7AC0>"
      ]
     },
     "execution_count": 97,

```
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "img = image.load_img(r\"E:\\Flask\\Sample_Images-20221104T061454Z-001\\Sample_Images\\Test_Image1.jpg\", grayscale=False,target_size= (64,64))\n",
   "img\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 98,
  "id": "8261cae3",
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "1/1 [==============================] - 0s 241ms/step\n"
    ]
   },
   {
    "data": {
     "text/plain": [
      "0"
     ]
    },
    "execution_count": 98,
    "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "x = image.img_to_array(img)\n",
  "x = np.expand_dims(x,axis = 0)\n",
  "pred = np.argmax(model.predict(x)) \n",
  "pred"
 ]
},
{
 "cell_type": "code",
 "execution_count": 99,
 "id": "3bd399b5",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "'APPLES'"
    ]
   },
   "execution_count": 99,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "labels=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']\n",
  "labels[np.argmax(pred)]"
 ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "d2d06946",
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "43b2d4b6",
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.9.12"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```