from keras.models import Sequential from keras.layers import Dense from keras.layers import Convolution2D from keras.layers import MaxPooling2D from keras.layers import Flatten

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale=1./255,shear\_range=0.2,zoom\_range=0.2,horizonta

test\_datagen = ImageDataGenerator(rescale=1./255)

x\_train = train\_datagen.flow\_from\_directory("/content/drive/MyDrive/AI\_IBM/Dataset/TRAIN\_S

Found 4119 images belonging to 5 classes. x\_test = test\_datagen.flow\_from\_directory("/content/drive/MyDrive/AI\_IBM/Dataset/TEST\_SET"

Found 929 images belonging to 5 classes.

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: 1355, 2: 1019, 3: 275, 4: 475})

model = Sequential()

model.add(Convolution2D(32,(3,3),input\_shape=(64,64,3),activation="relu"))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Convolution2D(32,(3,3),activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Flatten()) model.add(Dense(units=128,activation='relu'))

model.add(Dense(units=5,activation='softmax'))

model.add(Flatten()) model.summary()

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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 flatten\_1 (Flatten) (None, 5) 0

=================================================================

Total params: 813,733

Trainable params: 813,733

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

model.add(Dense(units=300,kernel\_initializer="random\_uniform",activation="relu")) model.add(Dense(units=200,kernel\_initializer="random\_uniform",activation="relu"))

model.add(Dense(units=5,kernel\_initializer="random\_uniform",activation="softmax")) len(x\_train)

129

model.add(Dense(units=128,activation="relu",kernel\_initializer="random\_uniform")) model.add(Dense(units=1,activation="sigmoid",kernel\_initializer="random\_uniform"))

model.compile(loss="binary\_crossentropy",optimizer="adam",metrics=['accuracy'])

model.fit\_generator(x\_train,steps\_per\_epoch=len(x\_train), validation\_data=x\_test, validati /usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:1: UserWarning: `Model.f

"""Entry point for launching an IPython kernel. Epoch 1/20

129/129 [==============================] - 42s 323ms/step - loss: -579.1954 - accurac Epoch 2/20

129/129 [==============================] - 35s 272ms/step - loss: -630.7393 - accurac Epoch 3/20

129/129 [==============================] - 35s 273ms/step - loss: -683.9399 - accurac Epoch 4/20

129/129 [==============================] - 35s 274ms/step - loss: -738.6011 - accurac

Epoch 5/20

129/129 [==============================] - 36s 275ms/step - loss: -795.0793 - accurac Epoch 6/20

129/129 [==============================] - 37s 286ms/step - loss: -853.5035 - accuracy Epoch 7/20

129/129 [==============================] - 36s 276ms/step - loss: -913.4440 - accuracy

Epoch 8/20

129/129 [==============================] - 36s 275ms/step - loss: -974.8712 - accuracy Epoch 9/20

129/129 [==============================] - 35s 274ms/step - loss: -1037.6532 - accuracy Epoch 10/20

129/129 [==============================] - 36s 275ms/step - loss: -1101.9432 - accuracy Epoch 11/20

129/129 [==============================] - 35s 273ms/step - loss: -1167.7832 - accuracy

Epoch 12/20

129/129 [==============================] - 35s 273ms/step - loss: -1235.0177 - accuracy Epoch 13/20

129/129 [==============================] - 35s 274ms/step - loss: -1303.9956 - accuracy Epoch 14/20

129/129 [==============================] - 35s 274ms/step - loss: -1374.5148 - accuracy

Epoch 15/20

129/129 [==============================] - 36s 276ms/step - loss: -1446.9734 - accuracy

Epoch 16/20

129/129 [==============================] - 35s 274ms/step - loss: -1520.6868 - accuracy Epoch 17/20

129/129 [==============================] - 35s 273ms/step - loss: -1596.1498 - accuracy Epoch 18/20

129/129 [==============================] - 35s 271ms/step - loss: -1673.0337 - accuracy

Epoch 19/20

129/129 [==============================] - 35s 273ms/step - loss: -1751.5466 - accuracy Epoch 20/20

129/129 [==============================] - 35s 270ms/step - loss: -1831.8647 - accura

<keras.callbacks.History at 0x7f60240c4c10>

model.save("nutrition.h5")

from tensorflow.keras.models import load\_model from keras.preprocessing import image model =load\_model("nutrition.h5")

import numpy as np

from tensorflow.keras.utils import load\_img from tensorflow.keras.utils import img\_to\_array img = load\_img(r'/content/drive/MyDrive/AI\_IBM/Nutrition Analysis Using Image Classificati x = img\_to\_array(img)

x= np.expand\_dims(x,axis = 0) predict\_x=model.predict(x)

classes\_x=np.argmax(predict\_x,axis = -1) classes\_x

1/1 [==============================] - 0s 424ms/step array([0])

index=['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON'] result=str(index[classes\_x[0]]) result

'APPLES'