Sprint-2

Model Building(Training,Saving,Testing the model)

|  |  |
| --- | --- |
| Date | 01 November 2022 |
| Team ID | Jonisha |
| Project Name | AI-powered Nutrition Analyzer for Fitness Enthusiasts |
| Maximum Marks |  |

**Dataset:**

* In our dataset we have collected images of the five variety of fruits.
* Apple
* Orange
* Pineapple
* Watermelon
* Banana

Drive link : <https://drive.google.com/file/d/1jzDjV7jYcIzlIieagaJdubMJ3YeLsry1/view?usp=share_link>

**Image Pre-processing:**

* Import The ImageDataGenerator Library
* Configure ImageDataGenerator Class
* Apply Image DataGenerator Functionality To Trainset And Testset

**Model Building:**

* Importing The Model Building Libraries
* Initializing The Model
* Adding CNN Layers
* Adding Dense Layers
* Configure The Learning Process
* Train the model
* Save the model
* Test the model

Data Collection

Download the dataset [here](https://drive.google.com/file/d/1jzDjV7jYcIzlIieagaJdubMJ3YeLsry1/view)

# Unzipping the dataset

!unzip '/content/Dataset.zip'

inflating: Dataset/TRAIN\_SET/WATERMELON/r\_288\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_289\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_28\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_290\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_291\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_292\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_293\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_294\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_295\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_296\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_297\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_298\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_299\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_29\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_2\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_300\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_301\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_302\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_303\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_304\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_305\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_306\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_307\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_308\_100.jpg

inflating: Dataset/TRAIN\_SET/WATERMELON/r\_309\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_30\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_310\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_311\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_312\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_313\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_314\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_315\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_31\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_32\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_33\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_34\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_35\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_36\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_37\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_38\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_39\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_3\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_40\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_41\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_42\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_43\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_44\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_45\_100.jpg

inflating: Dataset/TRAIN\_SET/WATERMELON/r\_46\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_4\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_50\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_57\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_5\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_6\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_7\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_81\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_8\_100.jpg inflating: Dataset/TRAIN\_SET/WATERMELON/r\_9\_100.jpg

Image Preprocessing

#Importing The ImageDataGenerator Library

from keras.preprocessing.image import ImageDataGenerator

Image Data Augmentation

#Configure ImageDataGenerator Class

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

Applying Image DataGenerator Functionality To Trainset And Testset

#Applying Image DataGenerator Functionality To Trainset And Testset x\_train = train\_datagen.flow\_from\_directory(

r'/content/Dataset/TRAIN\_SET',

target\_size=(64, 64),batch\_size=5,color\_mode='rgb',class\_mode='sparse') #Applying Image DataGenerator Functionality To Testset

x\_test = test\_datagen.flow\_from\_directory( r'/content/Dataset/TEST\_SET',

target\_size=(64, 64),batch\_size=5,color\_mode='rgb',class\_mode='sparse')

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

#checking the number of classes print(x\_train.class\_indices)

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

#checking the number of classes 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})

Model Building

* Importing The Model Building Libraries

import numpy as np import tensorflow as tf

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

* Initializing The Model

model = Sequential()

* Adding CNN Layers

# Initializing the CNN classifier = Sequential()

# First convolution layer and pooling

classifier.add(Conv2D(32, (3, 3), input\_shape=(64, 64, 3), activation='relu'))

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

# Second convolution layer and pooling classifier.add(Conv2D(32, (3, 3), activation='relu'))

# input\_shape is going to be the pooled feature maps from the previous convolution layer classifier.add(MaxPooling2D(pool\_size=(2, 2)))

# Flattening the layers classifier.add(Flatten())

* Adding Dense Layers

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

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

#summary of our model 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 2D) | (None, 14, 14, 32) | 0 |
| 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

# Compiling the CNN

# categorical\_crossentropy for more than 2

classifier.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['acc

* Train The Model

#Fitting the model

classifier.fit\_generator(generator=x\_train,steps\_per\_epoch = len(x\_train),epochs=20, valid

Epoch 1/20

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:2: UserWarning: `Model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 824/824 [==============================]  Epoch 2/20 | - 21s | 16ms/step | - loss: | 0.6172 | - accuracy: |
| 824/824 [==============================] | - 13s | 15ms/step | - loss: | 0.4115 | - accuracy: |
| Epoch 3/20  824/824 [==============================] | - 13s | 16ms/step | - loss: | 0.3766 | - accuracy: |
| Epoch 4/20  824/824 [==============================] | - 13s | 16ms/step | - loss: | 0.3484 | - accuracy: |
| Epoch 5/20 |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 824/824 [==============================]  Epoch 6/20 | - 13s | 16ms/step | - loss: | 0.3243 | - accuracy: |
| 824/824 [==============================] | - 13s | 16ms/step | - loss: | 0.3240 | - accuracy: |
| Epoch 7/20  824/824 [==============================] | - 13s | 16ms/step | - loss: | 0.2887 | - accuracy: |
| Epoch 8/20  824/824 [==============================] | - 13s | 16ms/step | - loss: | 0.2728 | - accuracy: |
| Epoch 9/20  824/824 [==============================] | - 13s | 16ms/step | - loss: | 0.2717 | - accuracy: |
| Epoch 10/20 |  |  |  |  |  |
| 824/824 [==============================] | - 14s | 17ms/step | - loss: | 0.2365 | - accuracy: |
| Epoch 11/20  824/824 [==============================] | - 13s | 15ms/step | - loss: | 0.2301 | - accuracy: |
| Epoch 12/20  824/824 [==============================] | - 13s | 15ms/step | - loss: | 0.2083 | - accuracy: |
| Epoch 13/20 |  |  |  |  |  |
| 824/824 [==============================] | - 13s | 15ms/step | - loss: | 0.2049 | - accuracy: |
| Epoch 14/20  824/824 [==============================] | - 12s | 15ms/step | - loss: | 0.1930 | - accuracy: |
| Epoch 15/20 |  |  |  |  |  |
| 824/824 [==============================] | - 13s | 15ms/step | - loss: | 0.1807 | - accuracy: |
| Epoch 16/20 |  |  |  |  |  |
| 824/824 [==============================] | - 13s | 15ms/step | - loss: | 0.1712 | - accuracy: |
| Epoch 17/20 |  |  |  |  |  |
| 824/824 [==============================] | - 13s | 15ms/step | - loss: | 0.1599 | - accuracy: |
| Epoch 18/20  824/824 [==============================] | - 13s | 15ms/step | - loss: | 0.1619 | - accuracy: |
| Epoch 19/20  824/824 [==============================] | - 13s | 15ms/step | - loss: | 0.1505 | - accuracy: |
| Epoch 20/20  824/824 [==============================] | - 12s | 15ms/step | - loss: | 0.1211 | - accuracy: |

<keras.callbacks.History at 0x7fd655833d90>

* Saving The Model

classifier.save('nutrition.h5')

* Testing The Model

#Predict the results

from tensorflow.keras.models import load\_model from keras.preprocessing import image

model = load\_model("nutrition.h5")

from tensorflow.keras.utils import img\_to\_array #loading of the image

img = load\_img(r'/content/Sample\_Images/Test\_Image1.jpg',grayscale=False,target\_size= (64, #image to array

x = img\_to\_array(img) #changing the shape

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 18ms/step

array([0])

index=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']

result=str(index[classes\_x[0]]) result

'APPLES'

[Colab HYPERLINK "https://colab.research.google.com/signup?utm\_source=footer&utm\_medium=link&utm\_campaign=footer\_links" HYPERLINK "https://colab.research.google.com/signup?utm\_source=footer&utm\_medium=link&utm\_campaign=footer\_links"paid HYPERLINK "https://colab.research.google.com/signup?utm\_source=footer&utm\_medium=link&utm\_campaign=footer\_links" HYPERLINK "https://colab.research.google.com/signup?utm\_source=footer&utm\_medium=link&utm\_campaign=footer\_links"products HYPERLINK "https://colab.research.google.com/signup?utm\_source=footer&utm\_medium=link&utm\_campaign=footer\_links"](https://colab.research.google.com/signup?utm_source=footer&utm_medium=link&utm_campaign=footer_links) - [Cancel contracts HYPERLINK "https://colab.research.google.com/cancel-subscription" HYPERLINK "https://colab.research.google.com/cancel-subscription"here](https://colab.research.google.com/cancel-subscription)