Team ID	PNT2022TMID33113
Project Name	AI - powered Nutrition Analyzer for Fitness Enthusiasts

AI_powered_Nutrition_Analyzer_for_Fitness_Enthusiasts

Model Building

```
[16]: # Importing Neccessary Libraries import numpy as np #used for
numerical analysis import tensorflow #open source used for both
ML and DL for computation from tensorflow.keras.models import
Sequential #it is a plain stack of Layers from tensorflow.keras
import layers #A Layer consists of a tensor-in tensor-out□
↪computation function

#Dense Layer is the regular deeply connected neural
network Layer from tensorflow.keras.layers import
Dense, Flatten
#Faltten-used fot flattening the input or change the
dimension from tensorflow.keras.layers import Conv2D,
MaxPooling2D, Dropout□ ↪#Convolutional layer
#MaxPooling2D-for downsampling the image
from keras.preprocessing.image import ImageDataGenerator
```

```
[17]: model=Sequential()
```

3.0.1 Creating the model

```
[18]: # 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())
```

```
[19]: classifier.add(Dense (units=128, activation='relu'))
      classifier.add(Dense (units=5, activation='softmax')) # softmax for more than 2
```

```
[20]: classifier.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D (None, 31,                0
 31, 32) )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling (None, 14, 14,            0
 32) 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

_________________________________________________________________
```

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

3.1 Fitting the model

```
[22]: classifier.fit_generator( generator=x_train, steps_per_epoch =
len(x_train),□
       ↪epochs=20, validation_data=x_test, validation_steps =
       len(x_test)) # No of□ ↪images in test set
```

```
Epoch 1/20
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
UserWarning: `Model.fit_generator` is deprecated and will be
removed in a future version.
Please use `Model.fit`, which supports
  generators. """Entry point for launching an
  IPython kernel.

526/526 [==============================] - 15s 13ms/step - loss:
0.1652 -
accuracy: 0.9391 - val_loss: 0.1162 - val_accuracy: 0.9621
Epoch 2/20
526/526 [==============================] - 6s 12ms/step - loss:
0.0592 -
accuracy: 0.9817 - val_loss: 0.0045 - val_accuracy: 1.0000
Epoch 3/20
526/526 [==============================] - 6s 12ms/step - loss:
4.5107e-04 -
accuracy: 1.0000 - val_loss: 0.0203 - val_accuracy: 0.9896
Epoch 4/20
526/526 [==============================] - 6s 12ms/step - loss:
1.8523e-04 -
accuracy: 1.0000 - val_loss: 0.0052 - val_accuracy: 1.0000
Epoch 5/20
526/526 [==============================] - 6s 12ms/step - loss:
8.9034e-05 -
accuracy: 1.0000 - val_loss: 0.0113 - val_accuracy: 0.9905
Epoch 6/20
526/526 [==============================] - 6s 12ms/step - loss:
5.6083e-05 -
accuracy: 1.0000 - val_loss: 0.0066 - val_accuracy: 1.0000
Epoch 7/20
526/526 [==============================] - 8s 14ms/step - loss:
3.1644e-05 -
accuracy: 1.0000 - val_loss: 0.0128 - val_accuracy: 0.9905
Epoch 8/20
526/526 [==============================] - 6s 12ms/step - loss:
2.3077e-05 -
accuracy: 1.0000 - val_loss: 0.0188 - val_accuracy: 0.9896
Epoch 9/20
526/526 [==============================] - 6s 12ms/step - loss:
2.8951e-05 -
accuracy: 1.0000 - val_loss: 0.0113 - val_accuracy: 0.9915
Epoch 10/20
```

```
526/526 [==============================] - 7s 13ms/step - loss:
1.6114e-05 accuracy: 1.0000 - val_loss: 0.0256 - val_accuracy:
0.9867
Epoch 11/20
526/526 [==============================] - 7s 13ms/step - loss:
1.4261e-05 accuracy: 1.0000 - val_loss: 0.0124 - val_accuracy:
0.9905
Epoch 12/20
526/526 [==============================] - 7s 13ms/step - loss:
5.8428e-06 accuracy: 1.0000 - val_loss: 0.0147 - val_accuracy:
0.9905
Epoch 13/20
526/526 [==============================] - 7s 13ms/step - loss:
4.0379e-06 accuracy: 1.0000 - val_loss: 0.0121 - val_accuracy:
0.9915
Epoch 14/20
526/526 [==============================] - 7s 13ms/step - loss:
4.0424e-06 -
accuracy: 1.0000 - val_loss: 0.0118 - val_accuracy: 0.9915
Epoch 15/20
526/526 [==============================] - 7s 13ms/step - loss:
2.0868e-06 accuracy: 1.0000 - val_loss: 0.0140 - val_accuracy:
0.9905
Epoch 16/20
526/526 [==============================] - 7s 13ms/step - loss:
1.3716e-06 -
accuracy: 1.0000 - val_loss: 0.0019 - val_accuracy: 1.0000
Epoch 17/20
526/526 [==============================] - 8s 14ms/step - loss:
1.5067e-06 -
accuracy: 1.0000 - val_loss: 0.0177 - val_accuracy: 0.9896
Epoch 18/20
526/526 [==============================] - 7s 13ms/step - loss:
1.2072e-06 -
accuracy: 1.0000 - val_loss: 0.0248 - val_accuracy: 0.9877
Epoch 19/20
526/526 [==============================] - 7s 13ms/step - loss:
7.0966e-07 -
accuracy: 1.0000 - val_loss: 0.0147 - val_accuracy: 0.9905
Epoch 20/20
526/526 [==============================] - 7s 13ms/step - loss:
0.0510 accuracy: 0.9890 - val_loss: 5.1513e-04 - val_accuracy:
1.0000
```

```
[22]: <keras.callbacks.History at 0x7f4ed0fb4c50>
```

```
[23]: classifier.save('nutrition.h5')
```

Test the model

```
[24]: ### Predicting our results from
 tensorflow.keras.preprocessing import
 image from tensorflow.keras.models
 import load_model #from
 keras.preprocessing import image
model = load_model("nutrition.h5") #Loading the model
 for testing

[25]: img = image.load_img('/content/TEST_SET/APPLES/151_100.jpg',
grayscale=False,
 ↪target_size= (64,64)) #Loading of the image
 #img =
 image.load_img('/content/dataset/Testing/bears/k4
 (88).
 ↪jpeg',target_size=(64,64)) x =
image.img_to_array(img)#image to array x =
np.expand_dims(x,axis=0) #changing the shape
 #pred = model.predict_classes(x) #predicting
 the classes
 #pred
 pred = np.argmax(model.predict(x))
 print(pred, model.predict(x))
```

```
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 18ms/step
0 [[1. 0. 0. 0. 0.]]
```

```
[26]: op = ['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON'] # Creating list
 ↪of output categories
 result = op[pred]
 print(result)
```

```
APPLES
```

```
[29]: img =
image.load_img('/content/TEST_SET/WATERMELON/143_100.jpg',
 ↪grayscale=False, target_size= (64,64)) #Loading of
 the image #img =
 image.load_img('/content/dataset/Testing/bears/k4
 (88).
 ↪jpeg',target_size=(64,64)) x =
image.img_to_array(img)#image to array x =
np.expand_dims(x,axis=0) #changing the shape
 #pred = model.predict_classes(x) #predicting
 the classes
```

```
#pred pred =
np.argmax(model.predict(x
)) print(pred,
model.predict(x))
```

```
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
4 [[0. 0. 0. 0. 1.]]
```

[30]:

```
op = ['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON'] # Creating list
 ↪of output categories
result = op[pred]
print(result)
```

```
WATERMELON
```