DATA COLLECTION AND PREPROCESSING

Nutrition Image Analysis using CNN

```
!unzip '/content/Dataset-Fruit.zip'
```

```
inflating: Dataset/TEST_SET/WATERMELON/r_221_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_222_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_223_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_224_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_225_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_226_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_227_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_228_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_229_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_22_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_230_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_231_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_232_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_233_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_234_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_235_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_236_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_237_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_238_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_239_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_23_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_240_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_241_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_242_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_243_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_244_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_24_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_25_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_26_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_27_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_288_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_289_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_28_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_290_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_291_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_292_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_293_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_294_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_295_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_296_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_297_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_298_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_299_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_29_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_300_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_301_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_302_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_303_100.jpg
```

```
inflating: Dataset/TEST_SET/WATERMELON/r_304_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_305_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_306_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_307_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_308_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_309_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_30_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_310_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_311_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_312_100.jpg
inflating: Dataset/TEST_SET/WATERMELON/r_313_100.jpg
```

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 computatic
#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
```

Image Data Agumentation

```
#setting parameter for Image Data agumentation to the training data
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizonta
#Image Data agumentation to the testing data
test_datagen=ImageDataGenerator(rescale=1./255)
```

Loading our data and performing data agumentation

```
#performing data agumentation to train data
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')
#performing data agumentation to test data
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 1500 images belonging to 5 classes.
```

```
print(x_train.class_indices)#checking the number of classes
```

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

```
print(x_test.class_indices)#checking the number of classes
```

```
{'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})
```

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

```
Counter({0: 266, 1: 415, 2: 248, 3: 224, 4: 347})
```

Creating the model

```
# 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 a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=5, activation='softmax')) # softmax for more than 2
```

```
classifier.summary()#summary of our model
```

```
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

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

Compiling the model

```
# Compiling the CNN
# categorical_crossentropy for more than 2
classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['acc
```

Fitting the model

```
classifier.fit_generator(
        generator=x_train,steps_per_epoch = len(x_train),
        epochs=10, validation_data=x_test,validation_steps = len(x_test))# No of images ir

Epoch 1/10
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.
  This is separate from the ipykernel package so we can avoid doing imports until
824/824 [==============================] - 23s 16ms/step - loss: 0.6025 - accuracy:
Epoch 2/10
824/824 [==============================] - 13s 16ms/step - loss: 0.4154 - accuracy:
Epoch 3/10
824/824 [==============================] - 13s 16ms/step - loss: 0.3692 - accuracy:
Epoch 4/10
824/824 [==============================] - 13s 16ms/step - loss: 0.3485 - accuracy:
Epoch 5/10
824/824 [==============================] - 13s 16ms/step - loss: 0.3162 - accuracy:
Epoch 6/10
824/824 [==============================] - 15s 18ms/step - loss: 0.3150 - accuracy:
Epoch 7/10
824/824 [==============================] - 13s 15ms/step - loss: 0.3039 - accuracy:
Epoch 8/10
824/824 [==============================] - 13s 15ms/step - loss: 0.2667 - accuracy:
Epoch 9/10
824/824 [==============================] - 13s 16ms/step - loss: 0.2597 - accuracy:
Epoch 10/10
824/824 [==============================] - 13s 16ms/step - loss: 0.2425 - accuracy:
<keras.callbacks.History at 0x7f57d22f6bd0>
```

Saving our model

```
# Save the model
classifier.save('nutrition.h5')
```

Nutrition Image Analysis using CNN

Predicting our results

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np


#test 1
img = image.load_img("/content/Dataset/TRAIN_SET/WATERMELON/127_100.jpg",target_size= (64,
img
```

```
x=image.img_to_array(img)#conversion image into array


x
```

```
array([[[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
```

```
       [255., 255., 255.],
       [255., 255., 255.]],

      ...,

      [[255., 255., 255.],
       [255., 255., 255.],
       [255., 255., 255.],
       ...,
       [255., 255., 255.],
       [255., 255., 255.],
       [255., 255., 255.]],

      [[255., 255., 255.],
       [255., 255., 255.],
       [255., 255., 255.],
       ...,
       [255., 255., 255.],
       [255., 255., 255.],
       [255., 255., 255.]],

      [[255., 255., 255.],
       [255., 255., 255.],
       [255., 255., 255.],
       ...,
       [255., 255., 255.],
       [255., 255., 255.],
       [255., 255., 255.]]], dtype=float32)
```

```
x.ndim
```

```
3
```

```
x=np.expand_dims(x,axis=0) #expand the dimension
```

```
x.ndim
```

```
4
```

```
pred = classifier.predict(x)
```

```
1/1 [==============================] - 0s 132ms/step
```

```
pred
```

```
array([[0., 0., 0., 0., 1.]], dtype=float32)
```

```
labels=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']
labels[np.argmax(pred)]
```

```
'WATERMELON'
```

```
#test 2
```

```
img = image.load_img("/content/Dataset/TEST_SET/APPLES/n07740461_1141.jpg",target_size= (6
img
```

```
x=image.img_to_array(img)#conversion image into array
```

```
x
```

```
array([[[20., 32., 32.],
        [15., 27., 27.],
        [13., 25., 23.],
        ...,
        [ 2., 11., 16.],
        [ 2.,  7., 11.],
        [ 0.,  4.,  7.]],

       [[24., 34., 33.],
        [13., 27., 28.],
        [17., 27., 29.],
        ...,
        [ 8., 26., 28.],
        [ 9., 24., 27.],
        [ 7., 23., 23.]],

       [[21., 35., 35.],
        [13., 27., 27.],
        [12., 26., 26.],
        ...,
        [ 9., 29., 38.],
        [14., 35., 40.],
        [12., 37., 41.]],

       ...,

       [[ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        ...,
        [98., 98., 98.],
        [ 6.,  6.,  6.],
        [ 0.,  0.,  0.]],

       [[ 0.,  0.,  0.],
        [ 1.,  0.,  0.],
        [ 0.,  1.,  0.],
        ...,
        [ 1.,  1.,  1.],
        [ 5.,  5.,  5.],
        [ 1.,  1.,  1.]],

       [[ 0.,  0.,  0.],
        [ 9.,  4.,  1.],
        [18.,  4.,  3.],
        ...,
```

```
               [ 1.,  2.,  4.],
               [ 1.,  1.,  1.],
               [ 2.,  2.,  2.]]], dtype=float32)
```

```
x.ndim
```

```
     3
```

```
x=np.expand_dims(x,axis=0) #expand the dimension
```

```
x.ndim
```

```
     4
```

```
pred = classifier.predict(x)
```

```
     1/1 [==============================] - 0s 15ms/step
```

```
pred
```

```
     array([[1., 0., 0., 0., 0.]], dtype=float32)
```

```
labels=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']
labels[np.argmax(pred)]
```

```
     'APPLES'
```