

## Project Development Phase

### Sprint 2

Date	10 November 2022
Team ID	PNT2022TMID42999
Project Name	AI-powered Nutrition Analyzer for Fitness Enthusiasts

### Loading and pre-processing the data:

#### Image pre-processing :

Import The ImageDataGenerator Library

Keras ImageDataGenerator class to perform data augmentation. Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.

```
[3] from keras.preprocessing.image import ImageDataGenerator
```

```
import numpy as np
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.models import Sequential
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
#Flatten-used for flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout #Convolutional layer
```

#### Define the parameters /arguments for ImageDataGenerator class:

The ImageDataGenerator transforms each image in the batch by a series of random translations, these translations are based on the arguments

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation

There are five main types of data augmentation techniques for image data; specifically:

- Image shifts via the width\_shift\_range and height\_shift\_range arguments.
- The image flips via the horizontal\_flip and vertical\_flip arguments.
- Image rotations via the rotation\_range argument □ Image brightness via the brightness\_range argument. □ Image zoom via the zoom\_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

### Applying ImageDataGenerator functionality to trainset and testset :

For Training set using flow\_from\_directory function.


Arguments:

- directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- batch\_size: Size of the batches of data. Default: 32.
- target\_size: Size to resize images after they are read from disk.
- class\_mode:
  - 'int': means that the labels are encoded as integers (e.g. for sparse\_categorical\_crossentropy loss).
  - 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical\_crossentropy loss).
  - 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary\_crossentropy).
  - None (no labels).


```
x_train=train_datagen.flow_from_directory(
    r'/content/drive/MyDrive/ibm_final/TRAIN_SET',
    target_size=(64,64),batch_size=32,class_mode='categorical')
x_test=train_datagen.flow_from_directory(
    r'/content/drive/MyDrive/ibm_final/TEST_SET',
    target_size=(64,64),batch_size=32,class_mode='categorical')
```

Found 2567 images belonging to 5 classes.  
Found 1055 images belonging to 5 classes.

Checking the number of classes in train and test data and counting the number of images in each class of train set data by using the counter function.

```
✓ 0s  print(x_train.class_indices)
{ 'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4 }

✓ 0s [9] print(x_test.class_indices)
{ 'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4 }

✓ 0s  from collections import Counter as c
c(x_train.labels)
Counter({0: 591, 1: 418, 2: 479, 3: 604, 4: 475})
```