### SPRINT-2

|  |  |
| --- | --- |
| DATE | 21 October 2022 |
| TEAM ID | PNT2022TMID30482 |
| PROJECT NAME | AI-powered Nutrition Analyzer for Fitness Enthusiasts |

### DATA COLLECTION

Collect images of different food items organized into subdirectories based on their respective names as shown in the project structure.

Create folders of types of food items that need to be recognized.

In this project, we have collected images of 5 types of food items apples, 'banana', 'orange', 'pineapple' and 'watermelon', they are saved in the respective subdirectories with their respective names.

For more accurate results we can collect images of high resolution and feed the model with more images.

You can download the dataset used in this project using the link below.

Data Set: [**LINK**](https://drive.google.com/drive/folders/1zpnSFRUQNazuPj95mSAIz0dLj-Ekk8AG)

Note: For better accuracy train on more images

**Image Preprocessing**

In this milestone, we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although performing some geometric transformations of images like rotation, scaling, translation, etc.

**click on the**[**Link**](https://thesmartbridge.com/documents/spsaimldocs/CNNprep.pdf)

**Import The ImageDataGenerator Library**

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.

Let us import the ImageDataGenerator class from Keras

https://lh4.googleusercontent.com/i2GGPoMi8FCdHVJ1XFdZ4ZdLv7cYXpPtX49hjv8EUIIufMLf-1YsnwaoW33m0kDUcWfi7QlBYLHYSu8yikyFfT7HAGOdu9iH9PcE2E-qeAVZ86zado6c_6hooaEoy_t_vJVDcong

**Configure ImageDataGenerator Class**

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.

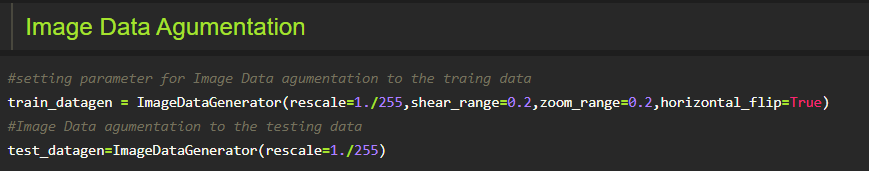
The image flips via the horizontal\_flip and vertical\_flip arguments.

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.



### Apply Image DataGenerator Functionality To Trainset And Testset

Let us apply ImageDataGenerator functionality to Trainset and Testset by using the following code

For Training set using flow\_from\_directory function.

This function will return batches of images from the subdirectories'apples', 'banana', 'orange', 'pineapple', 'watermelon' together with labels 0 to 4{'apples': 0, 'banana': 1, 'orange': 2, 'pineapple': 3, 'watermelon': 4}

Arguments:

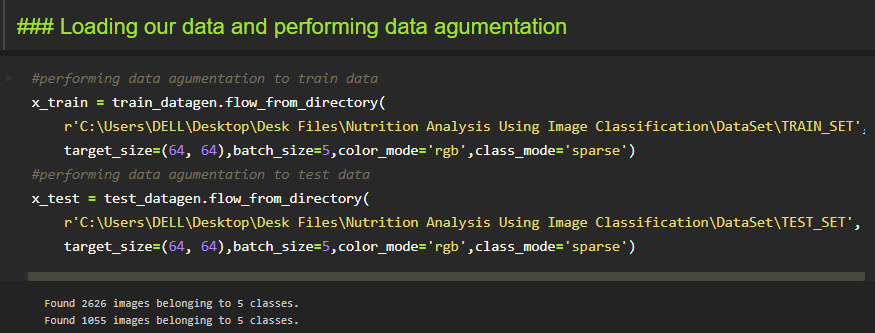
* directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
* batch\_size: Size of the batches of data. Default: 32.
* target\_size: Size to resize images after they are read from disk.
* class\_mode:

-  ‘int': means that the labels are encoded as integers (e.g. for sparse\_categorical\_crossentropy loss).

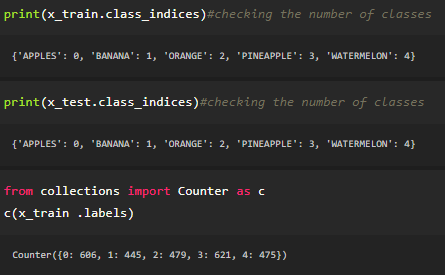
- 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical\_crossentropy loss).

- '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).



We notice that 2626 images are belonging to 5 classes for training and 1055 images belong to 5 classes for testing purposes.



Here we are 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.

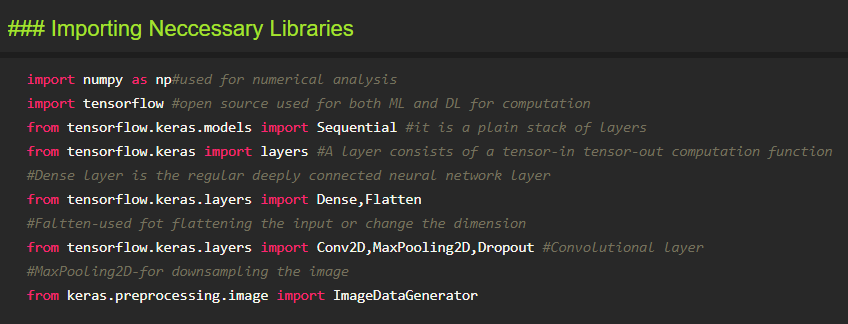
**Model Building**

Now it's time to build our Convolutional Neural Networking which contains an input layer along with the convolution, max-pooling, and finally an output layer.

* Link: [**https://thesmartbridge.com/documents/spsaimldocs/CNNflow.pdf**](https://thesmartbridge.com/documents/spsaimldocs/CNNflow.pdf)

**Importing The Model Building Libraries**

Importing the necessary libraries



**Initializing The Model**

Keras has 2 ways to define a neural network:

* Sequential
* Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add() method.

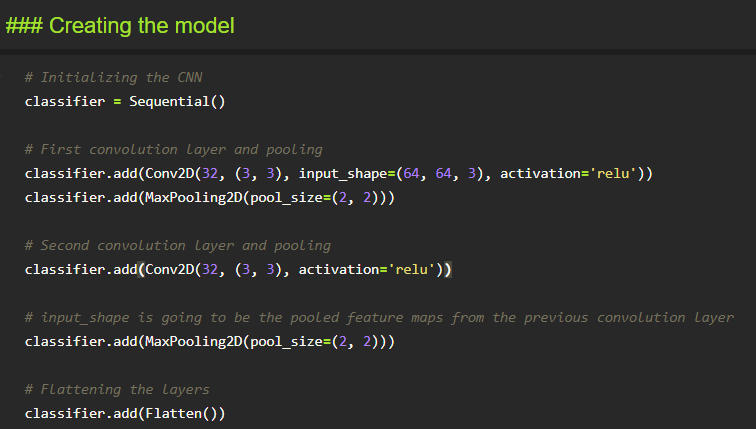
https://lh6.googleusercontent.com/M_VuUBnoT2LbCaW-ZPu4QzKYUnENczEmw67Zbn0wnYETGxW1UM9Sn4jRuyNUV0CQcOLSXveqjJXhMWSpr1bAS0M0Y5xkgSPVQQivLnFD0H9Z_Htgt_P5aivjxPXstMwgPswDYZvP

**Adding CNN Layers**

* For information regarding CNN Layers refer to the link Link: [**https://victorzhou.com/blog/intro-to-cnns-part-1/**](https://victorzhou.com/blog/intro-to-cnns-part-1/)
* As the input image contains three channels, we are specifying the input shape as (64,64,3).
* We are adding a two convolution layer with activation function as “relu” and

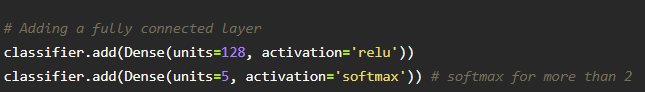
with a small filter size (3,3) and the number of filters (32) followed by a max-pooling layer.

* Max pool layer is used to down sample the input.(Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter)
* Flatten layer flattens the input. Does not affect the batch size.



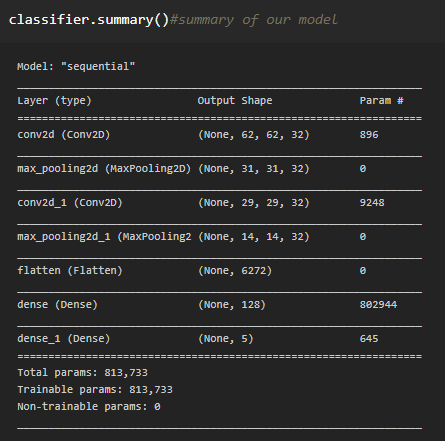
**Adding Dense Layers**

A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer.



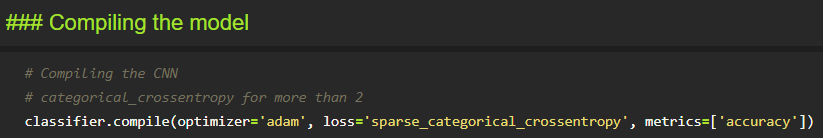
The number of neurons in the Dense layer is the same as the number of classes in the training set. The neurons in the last Dense layer, use softmax activation to convert their outputs into respective probabilities.

Understanding the model is a very important phase to properly using it for training and prediction purposes. Keras provides a simple method, a summary to get the full information about the model and its layers.



**Configure The Learning Process**

* The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires loss function during the model compilation process.
* Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer
* Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process

  
**Train The Model**

Now, let us train our model with our image dataset. The model is trained for 20 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 20 epochs and probably there is further scope to improve the model.

fit\_generator functions used to train a deep learning neural network

Arguments:

* steps\_per\_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of     steps\_per\_epoch as the total number of samples in your dataset divided by the batch size.
* Epochs: an integer and number of epochs we want to train our model for.
* validation\_data can be either:

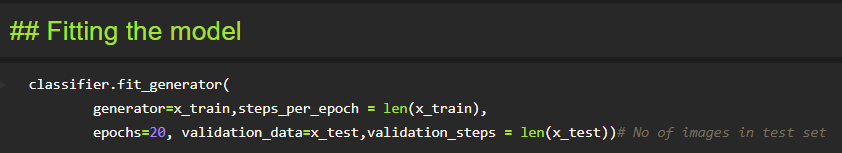
                      - an inputs and targets list

                      - a generator

                      - inputs, targets, and sample\_weights list which can be used to evaluate

                       -the loss and metrics for any model after any epoch has ended.

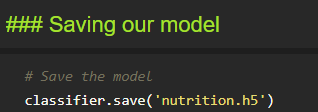
* validation\_steps: only if the validation\_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.



**Save The Model**

The model is saved with .h5 extension as follows

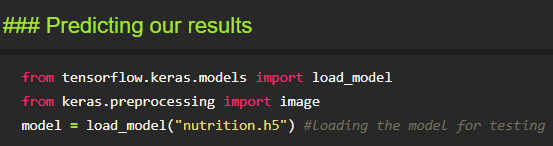
An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.



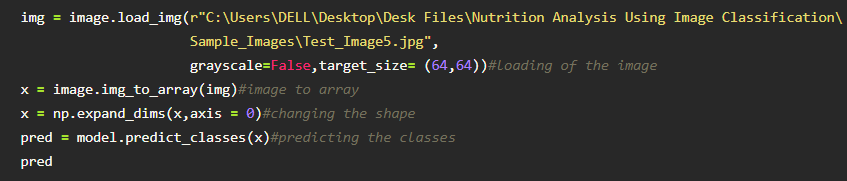
**Test The Model**

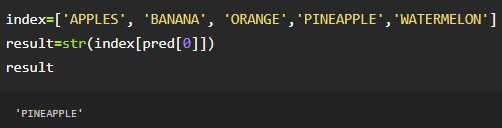
Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data.

Load the saved model using load\_model



Taking an image as input and checking the results

  
By using the model we are predicting the output for the given input image



The predicted class index name will be printed here.