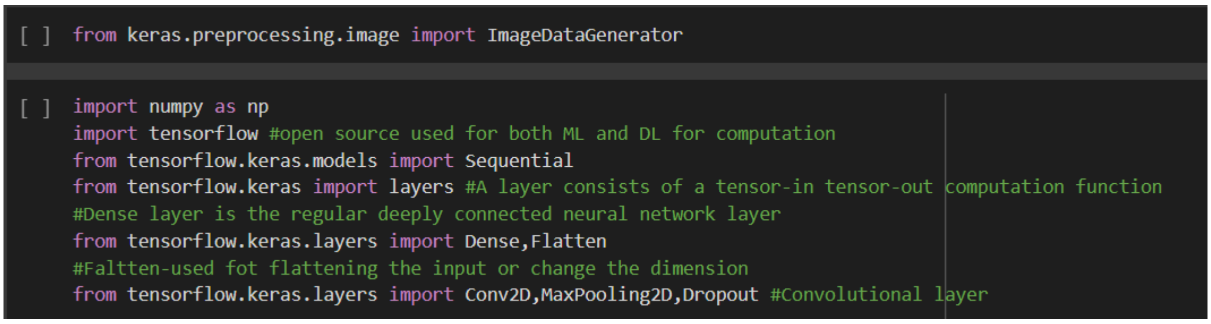
MODEL BUILDING

PROJECT TITLE : AI-powered Nutrition Analyzer for Fitness Enthusiasts

Team id : PNT2022TMID21516

Importing the model Building Libraries:

Importing the necessary libraries



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

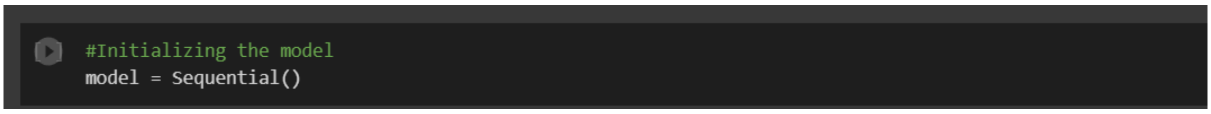
```
[ ]  import numpy as np
     import tensorflow #open source used for both ML and DL for computation
     from tensorflow.keras.models import Sequential
     from tensorflow.keras import layers #A layer consists of a tensor-in tensor-out computation function
     #Dense layer is the regular deeply connected neural network layer
     from tensorflow.keras.layers import Dense,Flatten
     #Faltten-used fot flattening the input or change the dimension
     from tensorflow.keras.layers import Conv2D,MaxPooling2D,Dropout #Convolutional layer
```

Initializing the model :

Keras has 2 ways to define a neural network:

- Sequential
- Function API

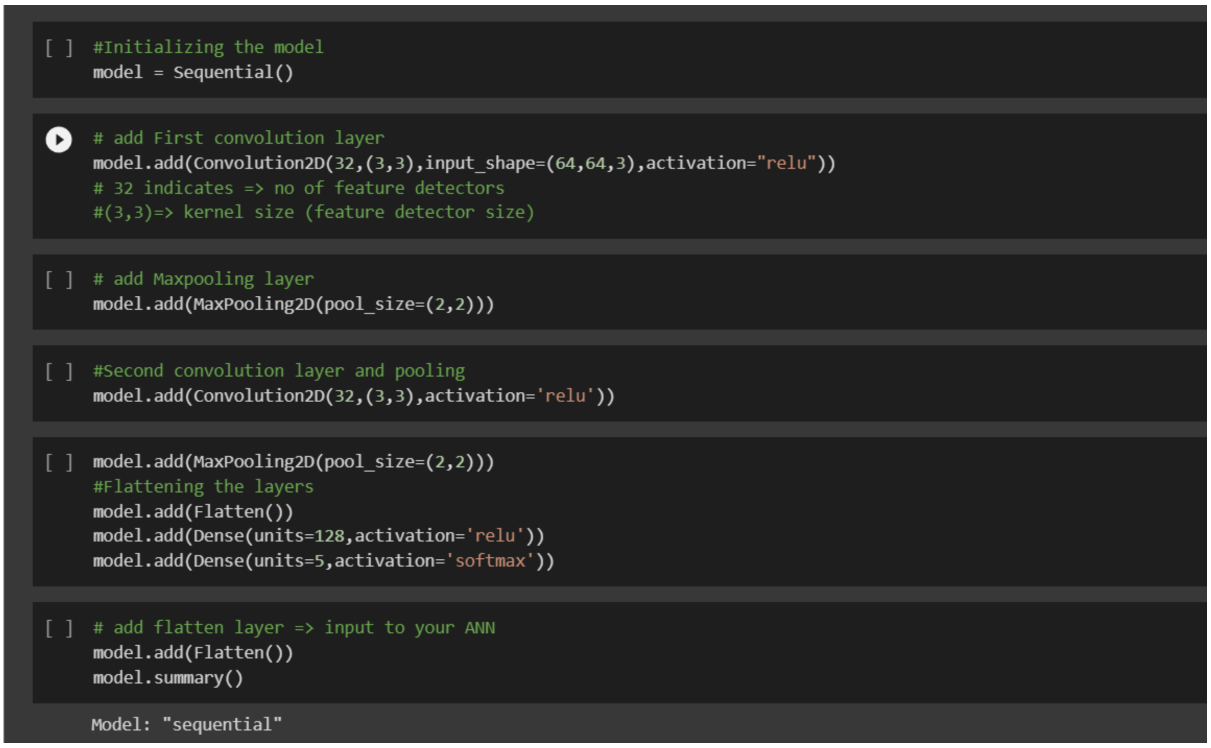
The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add() method.



```
#Initializing the model
model = Sequential()
```

Adding CNN layers:

- 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.



```
[ ]  #Initializing the model
     model = Sequential()
```

```
     # add First convolution layer
     model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation="relu"))
     # 32 indicates => no of feature detectors
     #(3,3)=> kernel size (feature detector size)
```

```
[ ]  # add Maxpooling layer
     model.add(MaxPooling2D(pool_size=(2,2)))
```

```
[ ]  #Second convolution layer and pooling
     model.add(Convolution2D(32,(3,3),activation='relu'))
```

```
[ ]  model.add(MaxPooling2D(pool_size=(2,2)))
     #Flattening the layers
     model.add(Flatten())
     model.add(Dense(units=128,activation='relu'))
     model.add(Dense(units=5,activation='softmax'))
```

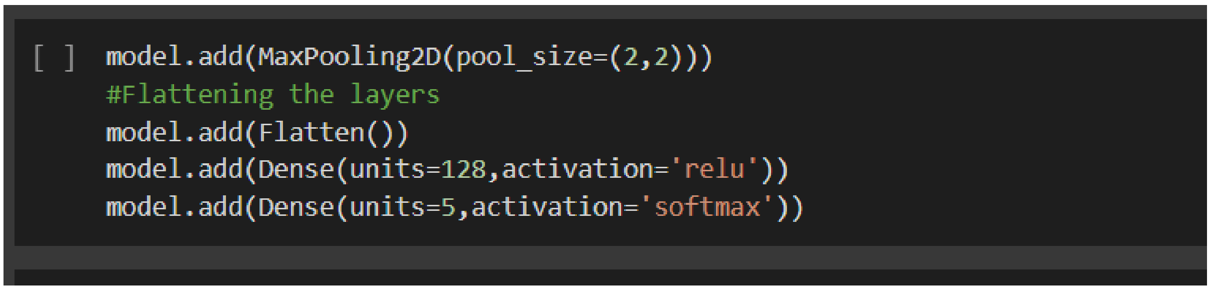
```
[ ]  # add flatten layer => input to your ANN
     model.add(Flatten())
     model.summary()

     Model: "sequential"
```

Adding dense layers:

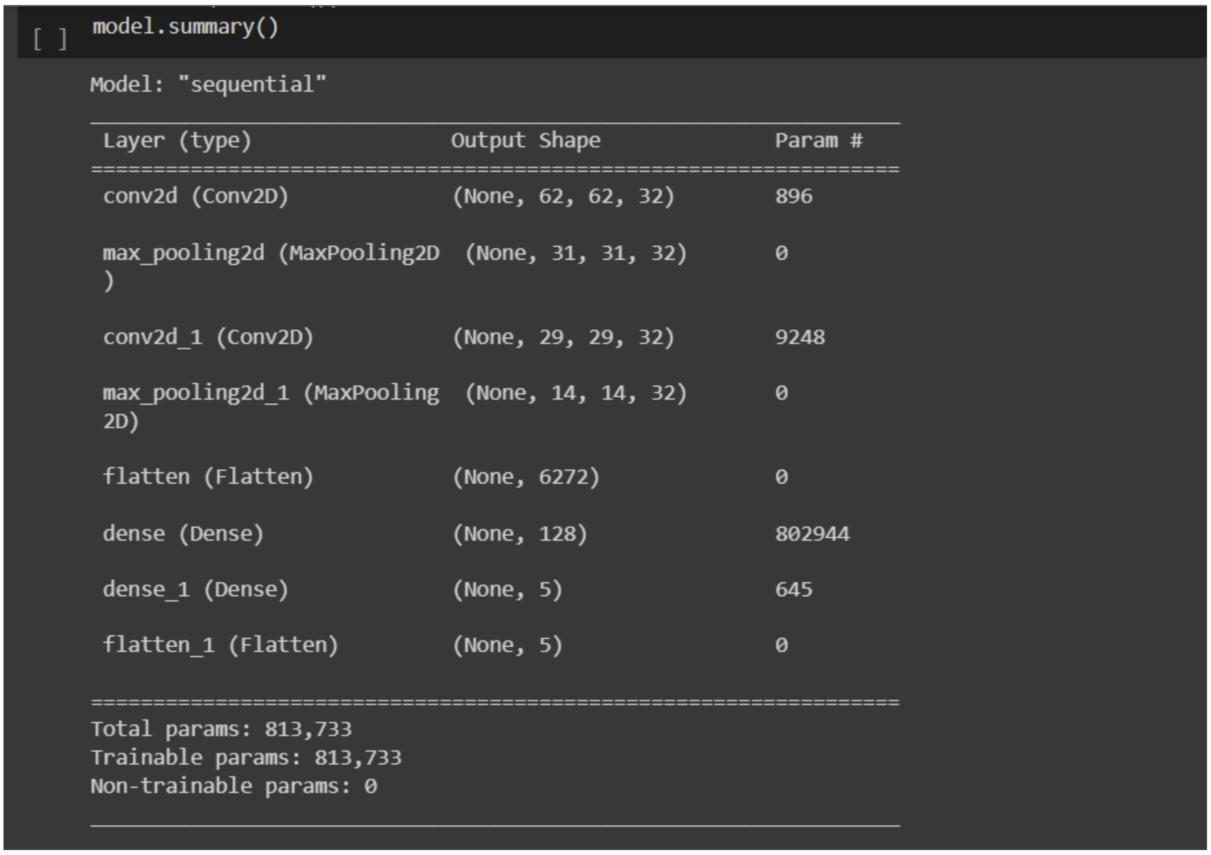
A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer.

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.



```
[ ]  model.add(MaxPooling2D(pool_size=(2,2)))
     #Flattening the layers
     model.add(Flatten())
     model.add(Dense(units=128,activation='relu'))
     model.add(Dense(units=5,activation='softmax'))
```



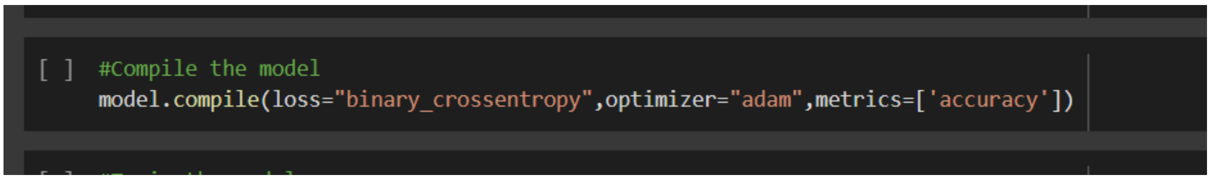
```
[ ] model.summary()

    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

     flatten_1 (Flatten)         (None, 5)                 0

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

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.

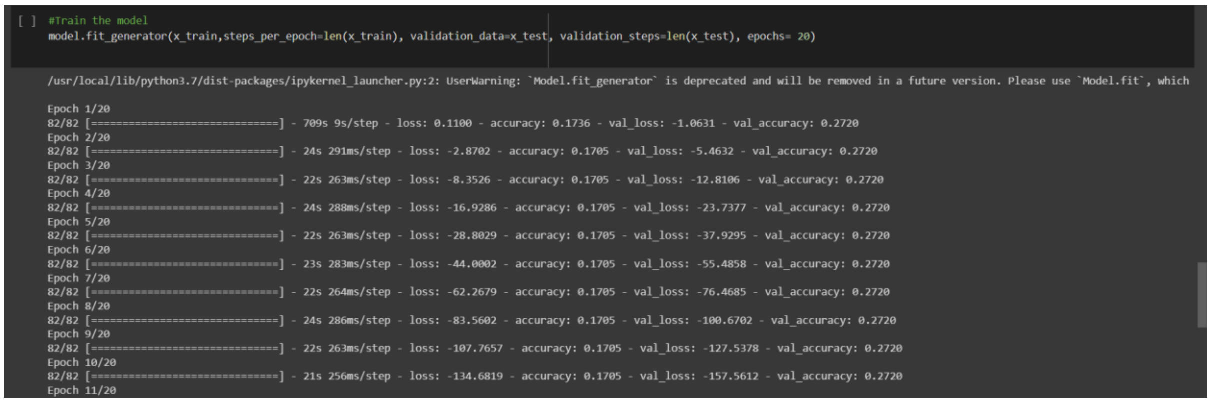
Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer

Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process



```
[ ] #Compile the model
    model.compile(loss="binary_crossentropy",optimizer="adam",metrics=['accuracy'])
```

Train the model :

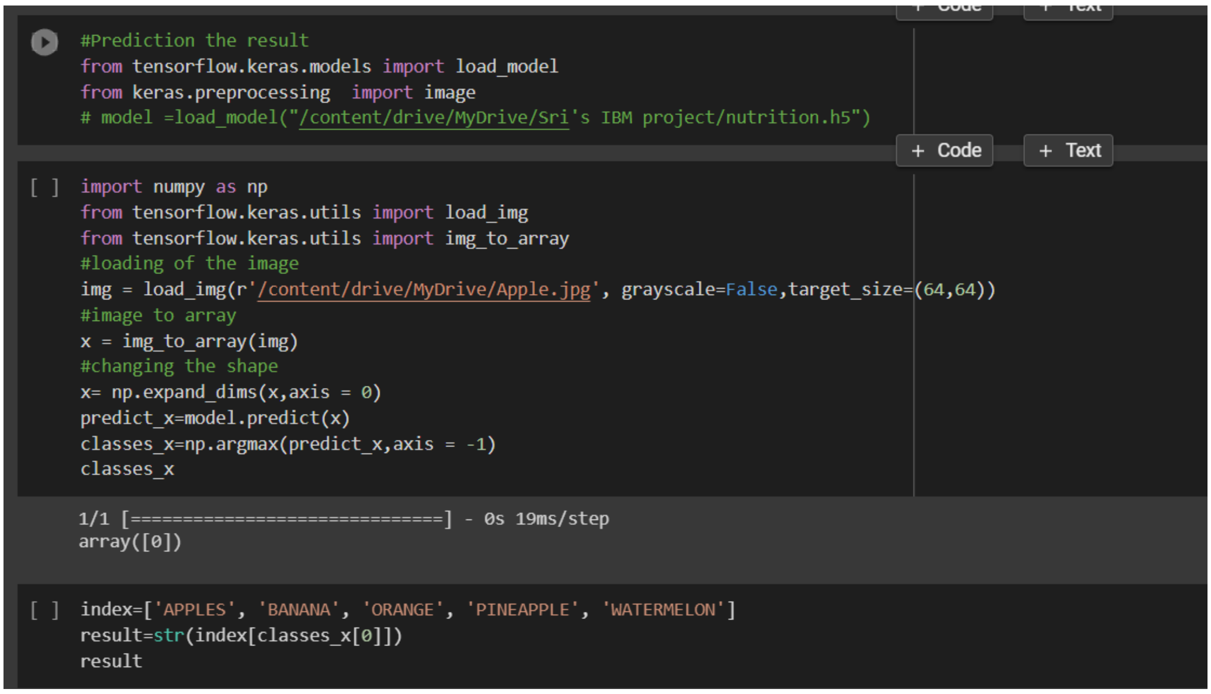


```
[ ] #Train the model
    model.fit_generator(x_train,steps_per_epoch=len(x_train), validation_data=x_test, validation_steps=len(x_test), epochs= 20)

    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which

    Epoch 1/20
    82/82 [==============================] - 709s 9s/step - loss: 0.1100 - accuracy: 0.1736 - val_loss: -1.0631 - val_accuracy: 0.2720
    Epoch 2/20
    82/82 [==============================] - 24s 291ms/step - loss: -2.8702 - accuracy: 0.1705 - val_loss: -5.4632 - val_accuracy: 0.2720
    Epoch 3/20
    82/82 [==============================] - 22s 263ms/step - loss: -8.3526 - accuracy: 0.1705 - val_loss: -12.8106 - val_accuracy: 0.2720
    Epoch 4/20
    82/82 [==============================] - 24s 288ms/step - loss: -16.9286 - accuracy: 0.1705 - val_loss: -23.7377 - val_accuracy: 0.2720
    Epoch 5/20
    82/82 [==============================] - 22s 263ms/step - loss: -28.8029 - accuracy: 0.1705 - val_loss: -37.9295 - val_accuracy: 0.2720
    Epoch 6/20
    82/82 [==============================] - 23s 283ms/step - loss: -44.0002 - accuracy: 0.1705 - val_loss: -55.4858 - val_accuracy: 0.2720
    Epoch 7/20
    82/82 [==============================] - 22s 264ms/step - loss: -62.2679 - accuracy: 0.1705 - val_loss: -76.4685 - val_accuracy: 0.2720
    Epoch 8/20
    82/82 [==============================] - 24s 286ms/step - loss: -83.5602 - accuracy: 0.1705 - val_loss: -100.6702 - val_accuracy: 0.2720
    Epoch 9/20
    82/82 [==============================] - 22s 263ms/step - loss: -107.7657 - accuracy: 0.1705 - val_loss: -127.5378 - val_accuracy: 0.2720
    Epoch 10/20
    82/82 [==============================] - 21s 256ms/step - loss: -134.6819 - accuracy: 0.1705 - val_loss: -157.5612 - val_accuracy: 0.2720
    Epoch 11/20
```

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.



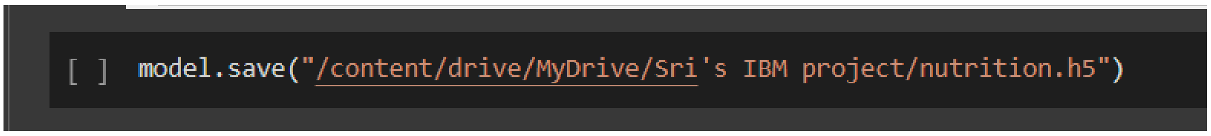
```
#Prediction the result
from tensorflow.keras.models import load_model
from keras.preprocessing  import image
# model =load_model("/content/drive/MyDrive/Sri's IBM project/nutrition.h5")
```

```
[ ] import numpy as np
    from tensorflow.keras.utils import load_img
    from tensorflow.keras.utils import img_to_array
    #loading of the image
    img = load_img(r'/content/drive/MyDrive/Apple.jpg', grayscale=False,target_size=(64,64))
    #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 19ms/step
    array([0])

[ ] index=['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON']
    result=str(index[classes_x[0]])
    result
```

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.



```
[ ]  model.save("/content/drive/MyDrive/Sri's IBM project/nutrition.h5")
```