

Topic: AI-powered Nutrition Analyzer for Fitness Enthusiasts

Team-ID: PNT2022TMID14614

GitHub ID: IBM-Project-31100-1660196130

Mentor Name: Mrs .Dhivya Kesavan

Team Members: Jemima Sharon.E

Mahalakshmi.R

Preethi.R

Preethi.S

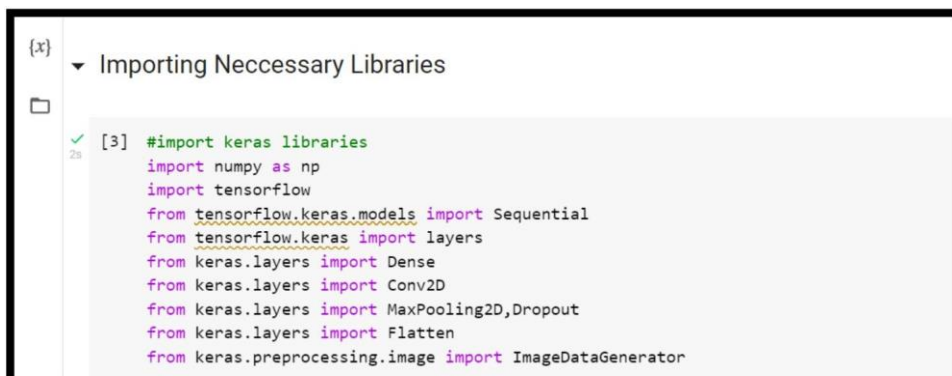
MODEL BUILDING

Here we are going to build our Convolutional Neural Networking which contains an input layer along with the convolution, max-pooling, and finally an output layer.

This is an important step because the model is the main thing needed for prediction.

- **Importing The Model Building Libraries**

We have imported all the necessary libraries needed for model building.



```
{x}
▼ Importing Neccessary Libraries

[3] #import keras libraries
import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from keras.layers import Dense
from keras.layers import Conv2D
from keras.layers import MaxPooling2D,Dropout
from keras.layers import Flatten
from keras.preprocessing.image import ImageDataGenerator
```

- **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. We will use the Sequential constructor to create our model, which will then have layers added to it using the add() method.

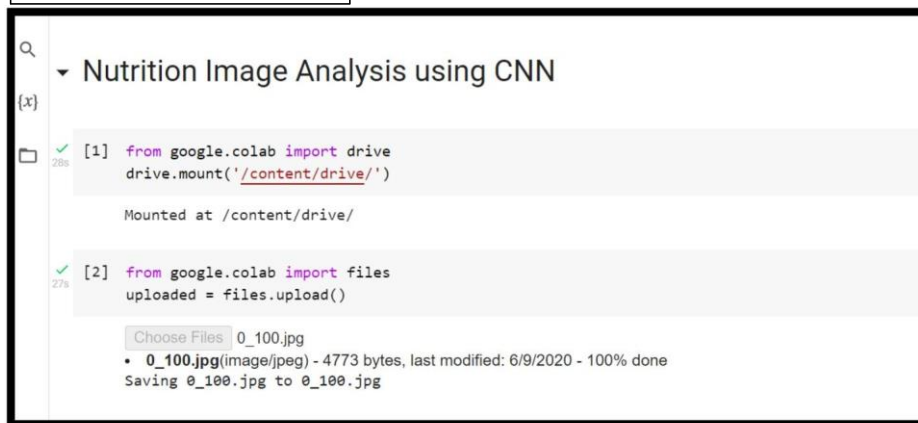
INITIALIZING MODEL

```
✓ [19] # Initializing the CNN  
3s classifier = 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 an 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 downsample 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 CNN LAYERS



```
[1] from google.colab import drive
drive.mount('/content/drive/')

Mounted at /content/drive/

[2] from google.colab import files
uploaded = files.upload()

Choose Files | 0_100.jpg
• 0_100.jpg(image/jpeg) - 4773 bytes, last modified: 6/9/2020 - 100% done
Saving 0_100.jpg to 0_100.jpg
```

- **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 use it for training and prediction purposes. Keras provides a simple method, and a summary to get the full information about the model and its layers.

ADDING DENSE LAYERS

```
MODEL SUMMARY

[19] # 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
```

- **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 a 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.

CONFIGURE THE LEARNING

```
Compiling the model

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

• Train The Model

Now, let us train our model with our image dataset. The model is trained for 15 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 15 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.

TRAIN THE MODEL

```
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

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: "Model.fit_generator" is deprecated and will be removed in a future version. Please use "Model.fit", which supports
This is separate from the ipykernel package so we can avoid doing imports until
Epoch 1/20
828/828 [=====] - 2278s 3s/step - loss: 0.6177 - accuracy: 0.7518 - val_loss: 0.3058 - val_accuracy: 0.8922
Epoch 2/20
828/828 [=====] - 23s 27ms/step - loss: 0.4231 - accuracy: 0.8432 - val_loss: 0.2778 - val_accuracy: 0.8940
Epoch 3/20
828/828 [=====] - 22s 26ms/step - loss: 0.3777 - accuracy: 0.8579 - val_loss: 0.2511 - val_accuracy: 0.9035
Epoch 4/20
828/828 [=====] - 21s 25ms/step - loss: 0.3469 - accuracy: 0.8685 - val_loss: 0.2343 - val_accuracy: 0.9136
Epoch 5/20
828/828 [=====] - 23s 27ms/step - loss: 0.3283 - accuracy: 0.8775 - val_loss: 0.2393 - val_accuracy: 0.9059
Epoch 6/20
828/828 [=====] - 22s 26ms/step - loss: 0.3087 - accuracy: 0.8840 - val_loss: 0.1993 - val_accuracy: 0.9279
Epoch 7/20
828/828 [=====] - 22s 26ms/step - loss: 0.2844 - accuracy: 0.8961 - val_loss: 0.2652 - val_accuracy: 0.9124
Epoch 8/20
828/828 [=====] - 21s 25ms/step - loss: 0.2726 - accuracy: 0.8978 - val_loss: 0.2293 - val_accuracy: 0.9172
Epoch 9/20
828/828 [=====] - 23s 27ms/step - loss: 0.2535 - accuracy: 0.9031 - val_loss: 0.2323 - val_accuracy: 0.9107
Epoch 10/20
828/828 [=====] - 22s 26ms/step - loss: 0.2343 - accuracy: 0.9111 - val_loss: 0.2075 - val_accuracy: 0.9256
Epoch 11/20
828/828 [=====] - 21s 25ms/step - loss: 0.2231 - accuracy: 0.9132 - val_loss: 0.2051 - val_accuracy: 0.9345
Epoch 12/20
828/828 [=====] - 22s 27ms/step - loss: 0.2075 - accuracy: 0.9251 - val_loss: 0.2178 - val_accuracy: 0.9232
Epoch 13/20
828/828 [=====] - 21s 26ms/step - loss: 0.1926 - accuracy: 0.9306 - val_loss: 0.1724 - val_accuracy: 0.9327
Epoch 14/20
✓ Os completed at 12:40 AM
```

- **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.

SAVE THE MODEL

▼ Saving Dataset.zip to Dataset.zip

```
[4] !unzip '/content/Dataset-20221118T061955Z-001.zip'
```

```
unzip: cannot find or open /content/Dataset-20221118T061955Z-001.zip, /content/Dataset-20221118T061955Z-001.zip.zip or /content/Dataset-20221118T061955Z-001.zip.zip.zip
```

- **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.

TEST THE MODEL

▼ Predicting our results

```
[24] from tensorflow.keras.models import load_model
      from keras.preprocessing import image
      from tensorflow.keras.preprocessing import image
      model = load_model("fruit.h5") #loading the model for testing
```

```
[29] img = tensorflow.keras.utils.load_img(r"/content/drive/My Drive/Sample Images/Test_Image1.jpg", grayscale=False, target_size=(64,64)) #loading of the image
      x = image.img_to_array(img) #image to array
      x = np.expand_dims(x, axis=0) #changing the shape
      #pred = (model.predict(x) > 0.5).astype("int32") #predicting the classes
      pred = np.argmax(model.predict(x), axis=-1)
      pred

1/1 [=====] - 0s 182ms/step
array([0])
```

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

'APPLES'
```