TRAINING THE MODEL ON IBM:

TEAM ID:PNT2022TMID50788

In [4]:
```
pwd
```

Out[4]:
```
'/home/wsuser/work'
```

In [15]:
```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='pFt6FEkmSn0nVp6zWde2pg6JkqDCb9Wp93_fqLqVnuw',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'imageprocessing-donotdelete-pr-joqstufszwpkdu'
object_key = 'Training-20221112T024940Z-001.zip'

streaming_body_3 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

In [ ]:
```
```

In [16]:
```
from io import BytesIO
```

In [16]:
```
from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_3.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

In [ ]:
```
ls
```

In [ ]:
```
pwd
```

In [17]:
```
from keras.preprocessing.image import ImageDataGenerator
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 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
#MaxPooling2D-for downsampling the image
from keras.preprocessing.image import ImageDataGenerator
```

In [18]:
```
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [19]:
```
x_train = train_datagen.flow_from_directory(
    r'/home/wsuser/work/Training/Dataset/TRAIN_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
x_test = test_datagen.flow_from_directory(
    r'/home/wsuser/work/Training/Dataset/TRAIN_SET',
```

In [19]:
```
x_train = train_datagen.flow_from_directory(
    r'/home/wsuser/work/Training/Dataset/TRAIN_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
x_test = test_datagen.flow_from_directory(
    r'/home/wsuser/work/Training/Dataset/TRAIN_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
```

```
Found 4138 images belonging to 5 classes.
Found 4138 images belonging to 5 classes.
```

In [20]:
```
print(x_train.class_indices)#checking the number of classes
```

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

In [21]:
```
print(x_test.class_indices) #checking the number of classes
```

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

In [22]:
```
from collections import Counter as c
c(x_train .labels)
```

Out[22]:
```
Counter({0: 995, 1: 1374, 2: 1019, 3: 275, 4: 475})
```

In [23]:
```
model=Sequential()
```

In [24]:
```
# Initializing the CNN classifier = Sequential()
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)))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Flattening the Layers
classifier.add(Flatten())
# Adding fully connected Layer a
classifier.add(Dense (units=128, activation='relu'))
classifier.add(Dense (units=5, activation='softmax')) # softmax for more than 2
```

In [25]:
```
classifier.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 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
_________________________________________________________________
```

In [26]:
```
classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

In [ ]:
```
##Fitting 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
```

```
/tmp/wsuser/ipykernel_165/2706448856.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Mode
l.fit`, which supports generators.
  classifier.fit_generator(
Epoch 1/20
828/828 [==============================] - 56s 67ms/step - loss: 0.6839 - accuracy: 0.7255 - val_loss: 0.4220 - val_accuracy: 0.8308
Epoch 2/20
828/828 [==============================] - 54s 66ms/step - loss: 0.4299 - accuracy: 0.8366 - val_loss: 0.3379 - val_accuracy: 0.8712
Epoch 3/20
828/828 [==============================] - 55s 66ms/step - loss: 0.3900 - accuracy: 0.8523 - val_loss: 0.3233 - val_accuracy: 0.8697
Epoch 4/20
828/828 [==============================] - 55s 66ms/step - loss: 0.3700 - accuracy: 0.8596 - val_loss: 0.3347 - val_accuracy: 0.8775
Epoch 5/20
828/828 [==============================] - 55s 66ms/step - loss: 0.3381 - accuracy: 0.8748 - val_loss: 0.3260 - val_accuracy: 0.8719
Epoch 6/20
828/828 [==============================] - 54s 65ms/step - loss: 0.3316 - accuracy: 0.8731 - val_loss: 0.2455 - val_accuracy: 0.9108
Epoch 7/20
828/828 [==============================] - 55s 66ms/step - loss: 0.3080 - accuracy: 0.8821 - val_loss: 0.2553 - val_accuracy: 0.9072
Epoch 8/20
828/828 [==============================] - 54s 65ms/step - loss: 0.2942 - accuracy: 0.8891 - val_loss: 0.2722 - val_accuracy: 0.8990
Epoch 9/20
828/828 [==============================] - 54s 66ms/step - loss: 0.2709 - accuracy: 0.8983 - val_loss: 0.2202 - val_accuracy: 0.9176
Epoch 10/20
828/828 [==============================] - 54s 65ms/step - loss: 0.2648 - accuracy: 0.8949 - val_loss: 0.2655 - val_accuracy: 0.8946
Epoch 11/20
828/828 [==============================] - 54s 66ms/step - loss: 0.2417 - accuracy: 0.9101 - val_loss: 0.2126 - val_accuracy: 0.9174
Epoch 12/20
828/828 [==============================] - 53s 65ms/step - loss: 0.2282 - accuracy: 0.9130 - val_loss: 0.2247 - val_accuracy: 0.9108
Epoch 13/20
828/828 [==============================] - 52s 63ms/step - loss: 0.2246 - accuracy: 0.9132 - val_loss: 0.2408 - val_accuracy: 0.9070
Epoch 14/20
828/828 [==============================] - 54s 65ms/step - loss: 0.2162 - accuracy: 0.9176 - val_loss: 0.1503 - val_accuracy: 0.9454
Epoch 15/20
828/828 [==============================] - 54s 65ms/step - loss: 0.1903 - accuracy: 0.9261 - val_loss: 0.1458 - val_accuracy: 0.9420
Epoch 16/20
Epoch 15/20
828/828 [==============================] - 54s 65ms/step - loss: 0.1903 - accuracy: 0.9261 - val_loss: 0.1458 - val_accuracy: 0.9420
Epoch 16/20
828/828 [==============================] - 54s 65ms/step - loss: 0.1776 - accuracy: 0.9316 - val_loss: 0.1147 - val_accuracy: 0.9526
Epoch 17/20
828/828 [==============================] - 54s 65ms/step - loss: 0.1696 - accuracy: 0.9357 - val_loss: 0.1248 - val_accuracy: 0.9553
Epoch 18/20
828/828 [==============================] - 54s 65ms/step - loss: 0.1477 - accuracy: 0.9442 - val_loss: 0.1842 - val_accuracy: 0.9321
Epoch 19/20
349/828 [===========>..................] - ETA: 24s - loss: 0.1653 - accuracy: 0.9323
```

In [ ]:
```
ls
```

In [ ]:
```
classifier.save('nutrition.h5')
```

In [ ]:
```
ls
```

In [ ]:
```
pwd
```

In [ ]:
```
### Predicting our results
from tensorflow.keras.models import load_model
from keras.preprocessing import image
model = load_model("nutrition.h5") #Loading the model for testing
from tensorflow.keras.preprocessing import image
```

In [ ]:
```
img =image.load_img(r"/home/wsuser/work/Training/Dataset/TRAIN_SET/ORANGE/100_100.jpg",grayscale=False, target_size= (64,64))
x = image.img_to_array(img)#image to array
x = np.expand_dims(x,axis=0) #changing the shape =
y =np.argmax(model.predict(x),axis=1)
index=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']
index[y[0]]
```

IBM Deployment

In [ ]:
```
!pip install watson-machine-learning-client
```

In [ ]:
```
!pip install keras==2.2.4
!pip install tensorflow==2.5.0
```

In [ ]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials={
    "url":"https://us-south.ml.cloud.ibm.com",
    "apikey":"i_teLwLmAcu6hFYEjyLnhQA-M7PIcL7Ps3_K8G3TyWrf"
}

client=APIClient(wml_credentials)
```

In [ ]:
```
client
```

In [ ]:
```
def guid_space_name(client,nutrition_deploy):
    space=client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']['name']==nutrition_deploy)['metadata']['id'])
```

Deployment

In [ ]:
```
space_uid=guid_space_name(client,'nutrition_deploy')
print("Space UID " + space_uid)
```

In [ ]:
```
client.set.default_space(space_uid)
```


```
client.set.default_space(space_uid)
```


In [ ]:
```
client.software_specifications.list(200)
```

In [ ]:
```
software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_rt22.1-py3.9')
```

In [ ]:
```
software_space_uid
```

In [ ]:
```
!tar -zcvf nutrition-classification-model.tgz nutrition.h5
```

In [ ]:
```
model_details=client.repository.store_model(model ='nutrition-classification-model.tgz', meta_props={
    client.repository.ModelMetaNames.NAME: "CNN Model Building",
    client.repository.ModelMetaNames.TYPE: 'tensorflow_2.7',
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_space_uid
})
```

In [ ]:
```
model_id = client.repository.get_model_id(model_details)
```

In [ ]:
```
model_id
```

In [ ]:
```
ls
```

In [ ]:
```
client.repository.download(model_id,'nutrition.tar.gb')
```

In [ ]:
```
pwd
```