

Model Development Phase Template

Date	22 April 2024
Team ID	738194
Project Title	RIPE-SENSE: MANGO QUALITY GRADING WITH IMAGE ANALYSIS AND DEEP LEARNING.
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshot.

Initial Model Training Code (5 marks):

VGG16:

```

17]: from tensorflow.keras.applications.vgg16 import VGG16
    from tensorflow.keras.layers import Dense, Flatten
    from tensorflow.keras.models import Model

29]: vgg = VGG16(include_top=False, input_shape=(128, 128, 3))
    vgg.trainable = False

30]: vgg.summary()

Model: "vgg16"

[37]: vgg16.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

[38]: history = vgg16.fit(train_generator, validation_data=validation_generator, epochs=100)

Epoch 92/100
18/18 [=====] - 32s 2s/step - loss: 0.0655 - accuracy: 0.9944 - val_loss: 0.6607 - val_accuracy: 0.7417
Epoch 93/100
18/18 [=====] - 33s 2s/step - loss: 0.0677 - accuracy: 0.9889 - val_loss: 0.5640 - val_accuracy: 0.7875
Epoch 94/100
18/18 [=====] - 31s 2s/step - loss: 0.0857 - accuracy: 0.9722 - val_loss: 0.7011 - val_accuracy: 0.7583
Epoch 95/100
18/18 [=====] - 30s 2s/step - loss: 0.0747 - accuracy: 0.9806 - val_loss: 0.5226 - val_accuracy: 0.7792
Epoch 96/100
18/18 [=====] - 31s 2s/step - loss: 0.0722 - accuracy: 0.9889 - val_loss: 0.6842 - val_accuracy: 0.7458
Epoch 97/100
18/18 [=====] - 31s 2s/step - loss: 0.0618 - accuracy: 0.9972 - val_loss: 0.6367 - val_accuracy: 0.7500
Epoch 98/100
18/18 [=====] - 30s 2s/step - loss: 0.0622 - accuracy: 0.9944 - val_loss: 0.6284 - val_accuracy: 0.7542
Epoch 99/100
18/18 [=====] - 31s 2s/step - loss: 0.0659 - accuracy: 0.9944 - val_loss: 0.6484 - val_accuracy: 0.7542
Epoch 100/100
18/18 [=====] - 32s 2s/step - loss: 0.0647 - accuracy: 0.9917 - val_loss: 0.5774 - val_accuracy: 0.7708

[47]: #testing accuracy on training dataset
    loss, accuracy = vgg16.evaluate(test_generator)
    print("Test Accuracy : {:.2f}%".format(accuracy*100))

30/30 [=====] - 27s 880ms/step - loss: 0.3550 - accuracy: 0.8767
Test Accuracy : 87.67%

[46]: #TESTING Accuracy on valisation dataset
    loss, accuracy = vgg16.evaluate(validation_generator)
    print("Validation Accuracy : {:.2f}%".format(accuracy*100))

12/12 [=====] - 11s 889ms/step - loss: 0.5621 - accuracy: 0.7833
Validation Accuracy : 78.33%

```

CNN (Sequential) :

```

: #importing the model building libraries
from tensorflow.keras import layers
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.initializers import glorot_uniform, glorot_normal, he_uniform
from sklearn.metrics import classification_report, confusion_matrix, precision_recall_curve
import seaborn as sns
from keras.models import Model
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import CategoricalCrossentropy
from tensorflow.keras.metrics import Accuracy

: model = Sequential()
model.add(Convolution2D((64),(3,3),kernel_initializer=glorot_uniform(seed=10),input_shape = (128,128,3),activation="relu"))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Flatten())
model.add(Dense(units=64,kernel_initializer=he_uniform(seed=10),activation='relu')) #first Hidden Layer
model.add(Dense(units=64,kernel_initializer=he_uniform(seed=10),activation='relu')) #second hidden Layer
model.add(Dense(units=3,kernel_initializer=he_uniform(seed=10),activation='softmax'))

C:\Users\Admin\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:99: UserWarning: Do not pass an `input_shape`/'input_dim' argument
to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(

: model.summary()

Model: "sequential"

13]: model.compile(optimizer = 'rmsprop',loss='categorical_crossentropy',metrics=["accuracy"])

14]: history = model.fit(train_generator,epochs =100, validation_data = validation_generator)

Epoch 92/100
24/24 — 16s 540ms/step - accuracy: 0.9305 - loss: 0.1885 - val_accuracy: 0.6917 - val_loss: 1.1080
Epoch 93/100
24/24 — 17s 598ms/step - accuracy: 0.9341 - loss: 0.1517 - val_accuracy: 0.5083 - val_loss: 3.2152
Epoch 94/100
24/24 — 16s 525ms/step - accuracy: 0.7848 - loss: 0.8802 - val_accuracy: 0.7417 - val_loss: 0.9215
Epoch 95/100
24/24 — 16s 532ms/step - accuracy: 0.9472 - loss: 0.1080 - val_accuracy: 0.6083 - val_loss: 1.4439
Epoch 96/100
24/24 — 16s 524ms/step - accuracy: 0.8901 - loss: 0.2393 - val_accuracy: 0.7000 - val_loss: 1.2387
Epoch 97/100
24/24 — 15s 514ms/step - accuracy: 0.9232 - loss: 0.2413 - val_accuracy: 0.6667 - val_loss: 1.3673
Epoch 98/100
24/24 — 15s 506ms/step - accuracy: 0.9594 - loss: 0.1133 - val_accuracy: 0.7167 - val_loss: 1.0998
Epoch 99/100
24/24 — 16s 546ms/step - accuracy: 0.9425 - loss: 0.1839 - val_accuracy: 0.6750 - val_loss: 1.1386
Epoch 100/100
24/24 — 16s 539ms/step - accuracy: 0.8976 - loss: 0.2280 - val_accuracy: 0.7417 - val_loss: 1.0512

18]: #TESTING Accuracy on valisation dataset
loss, accuracy = model.evaluate(validation_generator)
print("Validation Accuracy : {:.2f}%".format(accuracy*100))

6/6 — 3s 446ms/step - accuracy: 0.7385 - loss: 0.8158
Validation Accuracy : 72.50%

19]: #testing accuracy on training dataset
loss, accuracy = model.evaluate(test_generator)
print("Test Accuracy : {:.2f}%".format(accuracy*100))

30/30 — 8s 270ms/step - accuracy: 0.9445 - loss: 0.2349
Test Accuracy : 93.33%

```

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics																																																			
VGG16	<div>[36]: vgg16.summary() Model: "model_1"</div> <table><tr><th>Layer (type)</th><th>Output Shape</th><th>Param #</th></tr><tr><td>input_2 (InputLayer)</td><td>[(None, 128, 128, 3)]</td><td>0</td></tr><tr><td>block1_conv1 (Conv2D)</td><td>(None, 128, 128, 64)</td><td>1792</td></tr><tr><td>block1_conv2 (Conv2D)</td><td>(None, 128, 128, 64)</td><td>36928</td></tr><tr><td>block1_pool (MaxPooling2D)</td><td>(None, 64, 64, 64)</td><td>0</td></tr><tr><td>block2_conv1 (Conv2D)</td><td>(None, 64, 64, 128)</td><td>73856</td></tr><tr><td>block2_conv2 (Conv2D)</td><td>(None, 64, 64, 128)</td><td>147584</td></tr><tr><td>block2_pool (MaxPooling2D)</td><td>(None, 32, 32, 128)</td><td>0</td></tr><tr><td>block3_conv1 (Conv2D)</td><td>(None, 32, 32, 256)</td><td>295168</td></tr><tr><td>block3_conv2 (Conv2D)</td><td>(None, 32, 32, 256)</td><td>590080</td></tr><tr><td>block3_conv3 (Conv2D)</td><td>(None, 32, 32, 256)</td><td>590080</td></tr><tr><td>block3_pool (MaxPooling2D)</td><td>(None, 16, 16, 256)</td><td>0</td></tr><tr><td>block4_conv1 (Conv2D)</td><td>(None, 16, 16, 512)</td><td>1180160</td></tr><tr><td>block4_conv2 (Conv2D)</td><td>(None, 16, 16, 512)</td><td>2359808</td></tr><tr><td>block4_conv3 (Conv2D)</td><td>(None, 16, 16, 512)</td><td>2359808</td></tr><tr><td>block4_pool (MaxPooling2D)</td><td>(None, 8, 8, 512)</td><td>0</td></tr><tr><td>block5_conv1 (Conv2D)</td><td>(None, 8, 8, 512)</td><td>2359808</td></tr></table>	Layer (type)	Output Shape	Param #	input_2 (InputLayer)	[(None, 128, 128, 3)]	0	block1_conv1 (Conv2D)	(None, 128, 128, 64)	1792	block1_conv2 (Conv2D)	(None, 128, 128, 64)	36928	block1_pool (MaxPooling2D)	(None, 64, 64, 64)	0	block2_conv1 (Conv2D)	(None, 64, 64, 128)	73856	block2_conv2 (Conv2D)	(None, 64, 64, 128)	147584	block2_pool (MaxPooling2D)	(None, 32, 32, 128)	0	block3_conv1 (Conv2D)	(None, 32, 32, 256)	295168	block3_conv2 (Conv2D)	(None, 32, 32, 256)	590080	block3_conv3 (Conv2D)	(None, 32, 32, 256)	590080	block3_pool (MaxPooling2D)	(None, 16, 16, 256)	0	block4_conv1 (Conv2D)	(None, 16, 16, 512)	1180160	block4_conv2 (Conv2D)	(None, 16, 16, 512)	2359808	block4_conv3 (Conv2D)	(None, 16, 16, 512)	2359808	block4_pool (MaxPooling2D)	(None, 8, 8, 512)	0	block5_conv1 (Conv2D)	(None, 8, 8, 512)	2359808	<div>[37]: vgg16.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])</div> <div>[*]: history = vgg16.fit(train_generator,validation_data=validation_generator,epochs=100)</div> <div>Epoch 1/100 18/18 [=====] - 39s 2s/step - loss: 1.8731 - accuracy: 0.3972 - val_loss: 0.8497 - val_accuracy: 0.6883 Epoch 2/100 18/18 [=====] - 37s 2s/step - loss: 0.6974 - accuracy: 0.7472 - val_loss: 0.7806 - val_accuracy: 0.6625 Epoch 3/100 18/18 [=====] - 34s 2s/step - loss: 0.5450 - accuracy: 0.8222 - val_loss: 0.7515 - val_accuracy: 0.6583 Epoch 4/100 18/18 [=====] - 38s 2s/step - loss: 0.5181 - accuracy: 0.7694 - val_loss: 0.5467 - val_accuracy: 0.7917 Epoch 5/100 18/18 [=====] - 38s 2s/step - loss: 0.4441 - accuracy: 0.8583 - val_loss: 0.5923 - val_accuracy: 0.7417 Epoch 6/100</div>
Layer (type)	Output Shape	Param #																																																			
input_2 (InputLayer)	[(None, 128, 128, 3)]	0																																																			
block1_conv1 (Conv2D)	(None, 128, 128, 64)	1792																																																			
block1_conv2 (Conv2D)	(None, 128, 128, 64)	36928																																																			
block1_pool (MaxPooling2D)	(None, 64, 64, 64)	0																																																			
block2_conv1 (Conv2D)	(None, 64, 64, 128)	73856																																																			
block2_conv2 (Conv2D)	(None, 64, 64, 128)	147584																																																			
block2_pool (MaxPooling2D)	(None, 32, 32, 128)	0																																																			
block3_conv1 (Conv2D)	(None, 32, 32, 256)	295168																																																			
block3_conv2 (Conv2D)	(None, 32, 32, 256)	590080																																																			
block3_conv3 (Conv2D)	(None, 32, 32, 256)	590080																																																			
block3_pool (MaxPooling2D)	(None, 16, 16, 256)	0																																																			
block4_conv1 (Conv2D)	(None, 16, 16, 512)	1180160																																																			
block4_conv2 (Conv2D)	(None, 16, 16, 512)	2359808																																																			
block4_conv3 (Conv2D)	(None, 16, 16, 512)	2359808																																																			
block4_pool (MaxPooling2D)	(None, 8, 8, 512)	0																																																			
block5_conv1 (Conv2D)	(None, 8, 8, 512)	2359808																																																			
CNN (Sequential)	<div>model.summary() Model: "sequential"</div> <table><tr><th>Layer (type)</th><th>Output shape</th><th>Param #</th></tr><tr><td>conv2d (Conv2D)</td><td>(None, 126, 126, 64)</td><td>1,792</td></tr><tr><td>max_pooling2d (MaxPooling2D)</td><td>(None, 63, 63, 64)</td><td>0</td></tr><tr><td>flatten (Flatten)</td><td>(None, 254816)</td><td>0</td></tr><tr><td>dense (Dense)</td><td>(None, 64)</td><td>16,257,088</td></tr><tr><td>dense_1 (Dense)</td><td>(None, 64)</td><td>4,160</td></tr><tr><td>dense_2 (Dense)</td><td>(None, 3)</td><td>195</td></tr></table> <div>Total params: 16,263,235 (62.04 MB) Trainable params: 16,263,235 (62.04 MB) Non-trainable params: 0 (0.00 B)</div>	Layer (type)	Output shape	Param #	conv2d (Conv2D)	(None, 126, 126, 64)	1,792	max_pooling2d (MaxPooling2D)	(None, 63, 63, 64)	0	flatten (Flatten)	(None, 254816)	0	dense (Dense)	(None, 64)	16,257,088	dense_1 (Dense)	(None, 64)	4,160	dense_2 (Dense)	(None, 3)	195	<div>] history = model.fit(train_generator,epochs =100, validation_data = validation_generator)</div> <div>24/24 [=====] 15s 527ms/step - accuracy: 0.4869 - loss: 1.1389 - val_accuracy: 0.4417 - val_loss: 1.8673 Epoch 3/100 24/24 [=====] 15s 527ms/step - accuracy: 0.4885 - loss: 1.0827 - val_accuracy: 0.3417 - val_loss: 1.1816 Epoch 4/100 24/24 [=====] 15s 525ms/step - accuracy: 0.4568 - loss: 1.8568 - val_accuracy: 0.4583 - val_loss: 1.8885 Epoch 5/100 24/24 [=====] 15s 515ms/step - accuracy: 0.4585 - loss: 1.8298 - val_accuracy: 0.3333 - val_loss: 1.1195 Epoch 7/100 24/24 [=====] 15s 520ms/step - accuracy: 0.5395 - loss: 0.9871 - val_accuracy: 0.4083 - val_loss: 1.1422 24/24 [=====] Epoch 8/100 24/24 [=====] 15s 534ms/step - accuracy: 0.5451 - loss: 0.9451 - val_accuracy: 0.4833 - val_loss: 1.1988 Epoch 9/100 24/24 [=====] 15s 512ms/step - accuracy: 0.5569 - loss: 0.9757 - val_accuracy: 0.5083 - val_loss: 1.8838 Epoch 10/100 24/24 [=====] 15s 512ms/step - accuracy: 0.5427 - loss: 0.8884 - val_accuracy: 0.5417 - val_loss: 0.9935 Epoch 11/100</div>																														
Layer (type)	Output shape	Param #																																																			
conv2d (Conv2D)	(None, 126, 126, 64)	1,792																																																			
max_pooling2d (MaxPooling2D)	(None, 63, 63, 64)	0																																																			
flatten (Flatten)	(None, 254816)	0																																																			
dense (Dense)	(None, 64)	16,257,088																																																			
dense_1 (Dense)	(None, 64)	4,160																																																			
dense_2 (Dense)	(None, 3)	195																																																			

[37]: vgg16.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
[4]: history = vgg16.fit(train_generator,validation_data=validation_generator,epochs=100)

Epoch 1/100
18/18 [=====] - 39s 2s/step - loss: 1.0731 - accuracy: 0.3972 - val_loss: 0.9497 - val_accuracy: 0.6083
Epoch 2/100
18/18 [=====] - 37s 2s/step - loss: 0.6974 - accuracy: 0.7472 - val_loss: 0.7006 - val_accuracy: 0.6625
Epoch 3/100
18/18 [=====] - 34s 2s/step - loss: 0.5450 - accuracy: 0.8222 - val_loss: 0.7525 - val_accuracy: 0.6583
Epoch 4/100
18/18 [=====] - 38s 2s/step - loss: 0.5181 - accuracy: 0.7694 - val_loss: 0.5467 - val_accuracy: 0.7917
Epoch 5/100
18/18 [=====] - 38s 2s/step - loss: 0.4441 - accuracy: 0.8583 - val_loss: 0.5923 - val_accuracy: 0.7417
Epoch 6/100