Model Development Phase Template

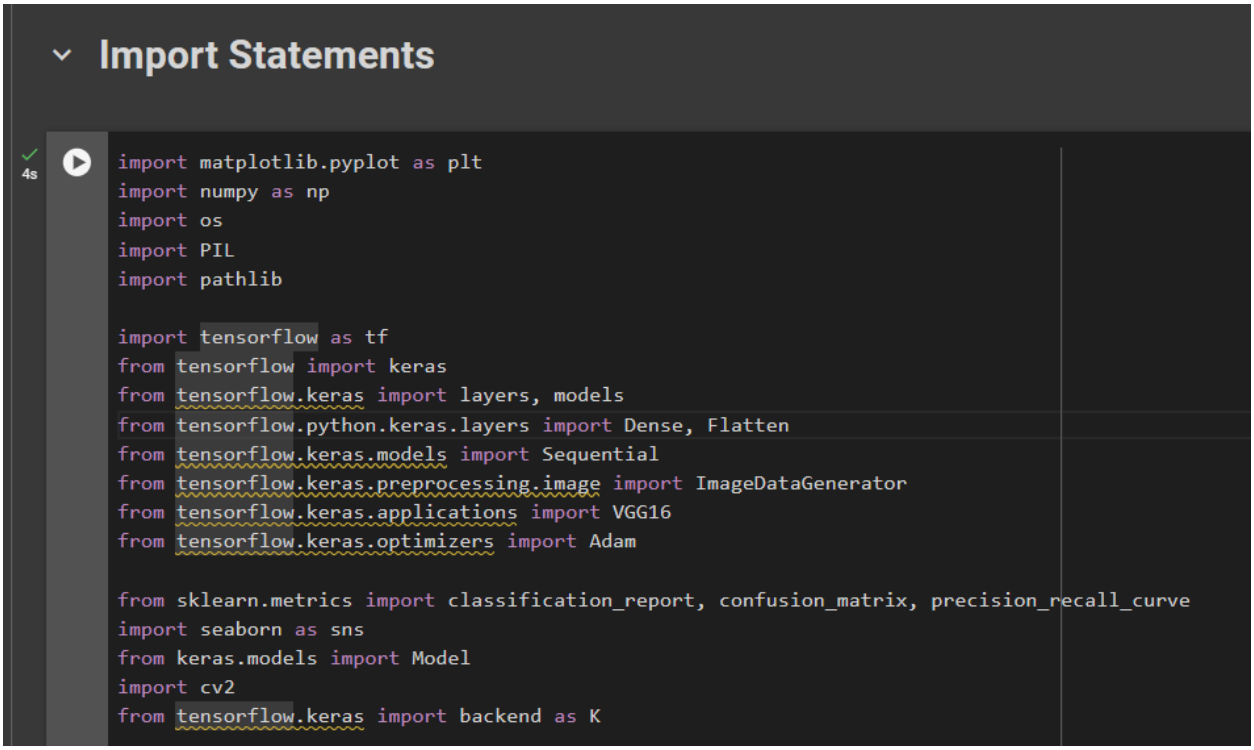
Date	11 July 2024
Team ID	SWTID1720080895
Project Title	RIPE-SENSE: MANGO QUALITY GRADING WITH IMAGE ANALYSIS AND DEEP LEARNING
Maximum Marks	10 Marks

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 screenshots.

Initial Model Training Code (5 marks):

VGG16:



VGG-16

```
[8]  vgg_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

     model = Sequential()
     for layer in vgg_model.layers:
         model.add(layer)

     for layer in model.layers:
         layer.trainable = False

     model.add(layers.Flatten())
     model.add(layers.Dense(512, activation='relu'))
     model.add(layers.BatchNormalization())
     model.add(layers.Dropout(0.5))
     model.add(layers.Dense(256, activation='relu'))
     model.add(layers.BatchNormalization())
     model.add(layers.Dropout(0.5))
     model.add(layers.Dense(3, activation='softmax'))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(
        train_generator,
        epochs=100,
        validation_data=validation_generator,
        verbose=1)
```

```
Epoch 72/100
15/15 [==============================] - 11s 736ms/step - loss: 0.3218 - accuracy: 0.8687 - val_loss: 0.5072 - val_accuracy: 0.7917
Epoch 73/100
15/15 [==============================] - 10s 703ms/step - loss: 0.2730 - accuracy: 0.8771 - val_loss: 0.5102 - val_accuracy: 0.8000
Epoch 74/100
15/15 [==============================] - 10s 654ms/step - loss: 0.2998 - accuracy: 0.8854 - val_loss: 0.3604 - val_accuracy: 0.8583
Epoch 75/100
15/15 [==============================] - 11s 734ms/step - loss: 0.2845 - accuracy: 0.8750 - val_loss: 0.4721 - val_accuracy: 0.7750
Epoch 76/100
15/15 [==============================] - 11s 735ms/step - loss: 0.2848 - accuracy: 0.8917 - val_loss: 0.5221 - val_accuracy: 0.7750
Epoch 77/100
15/15 [==============================] - 10s 686ms/step - loss: 0.2821 - accuracy: 0.8771 - val_loss: 0.4530 - val_accuracy: 0.8083
Epoch 78/100
15/15 [==============================] - 10s 675ms/step - loss: 0.3816 - accuracy: 0.8542 - val_loss: 0.5579 - val_accuracy: 0.7833
Epoch 79/100
15/15 [==============================] - 11s 734ms/step - loss: 0.3241 - accuracy: 0.8771 - val_loss: 0.6932 - val_accuracy: 0.7750
Epoch 80/100
15/15 [==============================] - 11s 745ms/step - loss: 0.3493 - accuracy: 0.8583 - val_loss: 0.5295 - val_accuracy: 0.7917
Epoch 81/100
15/15 [==============================] - 11s 723ms/step - loss: 0.3000 - accuracy: 0.8958 - val_loss: 0.5172 - val_accuracy: 0.7750
Epoch 82/100
15/15 [==============================] - 10s 645ms/step - loss: 0.2502 - accuracy: 0.8938 - val_loss: 0.6634 - val_accuracy: 0.7500
Epoch 83/100
15/15 [==============================] - 11s 712ms/step - loss: 0.3028 - accuracy: 0.8875 - val_loss: 0.8904 - val_accuracy: 0.7000
Epoch 84/100
15/15 [==============================] - 11s 728ms/step - loss: 0.3281 - accuracy: 0.8750 - val_loss: 1.0170 - val_accuracy: 0.6583
Epoch 85/100
15/15 [==============================] - 11s 743ms/step - loss: 0.2752 - accuracy: 0.8833 - val_loss: 0.8901 - val_accuracy: 0.6750
Epoch 86/100
15/15 [==============================] - 10s 643ms/step - loss: 0.2646 - accuracy: 0.8979 - val_loss: 1.0017 - val_accuracy: 0.6833
Epoch 87/100
15/15 [==============================] - 11s 729ms/step - loss: 0.2943 - accuracy: 0.8750 - val_loss: 0.7305 - val_accuracy: 0.6750
Epoch 88/100
15/15 [==============================] - 11s 736ms/step - loss: 0.3138 - accuracy: 0.8813 - val_loss: 0.6321 - val_accuracy: 0.7167
Epoch 89/100
15/15 [==============================] - 11s 721ms/step - loss: 0.3474 - accuracy: 0.8458 - val_loss: 0.4462 - val_accuracy: 0.8000
Epoch 90/100
15/15 [==============================] - 10s 680ms/step - loss: 0.3210 - accuracy: 0.8604 - val_loss: 0.4023 - val_accuracy: 0.8583
Epoch 91/100
15/15 [==============================] - 10s 684ms/step - loss: 0.2548 - accuracy: 0.8917 - val_loss: 0.4962 - val_accuracy: 0.7917
Epoch 92/100
15/15 [==============================] - 11s 738ms/step - loss: 0.3232 - accuracy: 0.8396 - val_loss: 0.6341 - val_accuracy: 0.7167
Epoch 93/100
15/15 [==============================] - 11s 732ms/step - loss: 0.2915 - accuracy: 0.8875 - val_loss: 2.6350 - val_accuracy: 0.3667
Epoch 94/100
15/15 [==============================] - 10s 649ms/step - loss: 0.2933 - accuracy: 0.8917 - val_loss: 0.3751 - val_accuracy: 0.8500
Epoch 95/100
15/15 [==============================] - 11s 689ms/step - loss: 0.2913 - accuracy: 0.8938 - val_loss: 0.5466 - val_accuracy: 0.7917
Epoch 96/100
15/15 [==============================] - 11s 731ms/step - loss: 0.2530 - accuracy: 0.9187 - val_loss: 0.4572 - val_accuracy: 0.8167
Epoch 97/100
15/15 [==============================] - 11s 727ms/step - loss: 0.2833 - accuracy: 0.8729 - val_loss: 0.4223 - val_accuracy: 0.8417
Epoch 98/100
15/15 [==============================] - 11s 734ms/step - loss: 0.2517 - accuracy: 0.9042 - val_loss: 0.6970 - val_accuracy: 0.7167
Epoch 99/100
15/15 [==============================] - 10s 657ms/step - loss: 0.2988 - accuracy: 0.8938 - val_loss: 0.7980 - val_accuracy: 0.7083
Epoch 100/100
15/15 [==============================] - 11s 700ms/step - loss: 0.2828 - accuracy: 0.8938 - val_loss: 0.7142 - val_accuracy: 0.7750
```

Validation Accuracy

```
loss, accuracy = model.evaluate(validation_generator)
print('Validation Accuracy: {:.2f}%'.format(accuracy*100))
```

```
4/4 [==============================] - 2s 498ms/step - loss: 0.6331 - accuracy: 0.8000
Validation Accuracy: 80.00%
```

Testing Accuracy

```
loss, accuracy = model.evaluate(test_generator)
print('Test Accuracy: {:.2f}%'.format(accuracy*100))
```

```
19/19 [==============================] - 4s 196ms/step - loss: 0.3938 - accuracy: 0.9284
Test Accuracy: 92.84%
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
```

```
flatten (Flatten)           (None, 25088)             0

dense (Dense)               (None, 512)               12845568

batch_normalization (Batch  (None, 512)               2048
Normalization)

dropout (Dropout)           (None, 512)               0

dense_1 (Dense)             (None, 256)               131328

batch_normalization_1 (Bat  (None, 256)               1024
chNormalization)

dropout_1 (Dropout)         (None, 256)               0

dense_2 (Dense)             (None, 3)                 771

=================================================================
Total params: 27695427 (105.65 MB)
Trainable params: 12979203 (49.51 MB)
Non-trainable params: 14716224 (56.14 MB)
_________________________________________________________________
```

EfficientNet:

EfficientNet

```
[26] import os
     import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras import regularizers, layers, Model
     from tensorflow.keras.applications import EfficientNetB2
     from tensorflow.keras.optimizers import Adamax
     from keras.callbacks import ReduceLROnPlateau, EarlyStopping


     sdir = '/content/drive/MyDrive/Grading_dataset'
     classlist = os.listdir(sdir)
     filepaths = []
     labels = []
     for c in classlist:
         classpath = os.path.join(sdir, c)
         for f in os.listdir(classpath):
             filepaths.append(os.path.join(classpath, f))
             labels.append(c)

     df = pd.DataFrame({'filepaths': filepaths, 'labels': labels})
```

```
trsplit = 0.8
vsplit = 0.1
dsplit = vsplit / (1 - trsplit)
train_df, dummy_df = train_test_split(df, train_size=trsplit, shuffle=True, random_state=123)
valid_df, test_df = train_test_split(dummy_df, train_size=dsplit, shuffle=True, random_state=123)

print(f'train_df length: {len(train_df)}, test_df length: {len(test_df)}, valid_df length: {len(valid_df)}')
print(f'Class distribution in training set: {train_df["labels"].value_counts().values}')

height, width, channels = 224, 224, 3
batch_size = 30
img_size = (height, width)
length = len(test_df)
test_batch_size = max([int(length / n) for n in range(1, length + 1) if length % n == 0 and length / n <= 80])
test_steps = int(length / test_batch_size)
print(f'test batch size: {test_batch_size}, test steps: {test_steps}')
```

```
train_df length: 480, test_df length: 60, valid_df length: 60
Class distribution in training set: [163 159 158]
test batch size: 60, test steps: 1
```

```
def scalar(img): return img

trgen = ImageDataGenerator(preprocessing_function=scalar, horizontal_flip=True)
tvgen = ImageDataGenerator(preprocessing_function=scalar)
train_gen = trgen.flow_from_dataframe(train_df, x_col='filepaths', y_col='labels', target_size=img_size,
                                      class_mode='categorical', color_mode='rgb', shuffle=True, batch_size=batch_size)
test_gen = tvgen.flow_from_dataframe(test_df, x_col='filepaths', y_col='labels', target_size=img_size,
                                     class_mode='categorical', color_mode='rgb', shuffle=False, batch_size=test_batch_size)
valid_gen = tvgen.flow_from_dataframe(valid_df, x_col='filepaths', y_col='labels', target_size=img_size,
                                      class_mode='categorical', color_mode='rgb', shuffle=True, batch_size=batch_size)
classes = list(train_gen.class_indices.keys())
class_count = len(classes)
train_steps = np.ceil(len(train_gen.labels) / batch_size)

base_model = EfficientNetB2(include_top=False, weights="imagenet", input_shape=(height, width, channels), pooling='max')
x = layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001)(base_model.output)
x = layers.Dense(256, kernel_regularizer=regularizers.l2(0.016), activity_regularizer=regularizers.l1(0.006),
                 bias_regularizer=regularizers.l1(0.006), activation='relu')(x)
x = layers.Dropout(rate=0.45, seed=123)(x)
output = layers.Dense(class_count, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=output)
model.compile(optimizer=Adamax(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
Found 480 validated image filenames belonging to 3 classes.
Found 60 validated image filenames belonging to 3 classes.
Found 60 validated image filenames belonging to 3 classes.
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb2_notop.h5
31790344/31790344 [==============================] - 1s 0us/step
```

```
epochs = 100
callbacks = [
    ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=1, verbose=1),
    EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True, verbose=1)
]
history = model.fit(train_gen, epochs=epochs, verbose=1, callbacks=callbacks, validation_data=valid_gen,
                    validation_steps=None, shuffle=False, initial_epoch=0)
```

```
Epoch 60/100
16/16 [==============================] - 7s 408ms/step - loss: 0.3186 - accuracy: 1.0000 - val_loss: 0.3338 - val_accuracy: 0.9833 - lr: 5.0000e-04
Epoch 61/100
16/16 [==============================] - 7s 399ms/step - loss: 0.3100 - accuracy: 0.9979 - val_loss: 0.3253 - val_accuracy: 0.9833 - lr: 5.0000e-04
Epoch 62/100
16/16 [==============================] - 7s 414ms/step - loss: 0.3029 - accuracy: 0.9979 - val_loss: 0.3210 - val_accuracy: 0.9833 - lr: 5.0000e-04
Epoch 63/100
16/16 [==============================] - 7s 401ms/step - loss: 0.3033 - accuracy: 0.9958 - val_loss: 0.3207 - val_accuracy: 0.9833 - lr: 5.0000e-04
Epoch 64/100
16/16 [==============================] - 7s 404ms/step - loss: 0.2955 - accuracy: 0.9979 - val_loss: 0.3030 - val_accuracy: 0.9833 - lr: 5.0000e-04
Epoch 65/100
16/16 [==============================] - ETA: 0s - loss: 0.2900 - accuracy: 0.9979
Epoch 65: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
16/16 [==============================] - 7s 400ms/step - loss: 0.2900 - accuracy: 0.9979 - val_loss: 0.3111 - val_accuracy: 0.9833 - lr: 5.0000e-04
Epoch 66/100
16/16 [==============================] - 7s 400ms/step - loss: 0.2832 - accuracy: 0.9958 - val_loss: 0.3028 - val_accuracy: 0.9833 - lr: 2.5000e-04
Epoch 67/100
16/16 [==============================] - 7s 415ms/step - loss: 0.2756 - accuracy: 1.0000 - val_loss: 0.2982 - val_accuracy: 0.9833 - lr: 2.5000e-04
Epoch 68/100
16/16 [==============================] - 6s 398ms/step - loss: 0.2781 - accuracy: 0.9979 - val_loss: 0.2956 - val_accuracy: 0.9833 - lr: 2.5000e-04
Epoch 69/100
16/16 [==============================] - ETA: 0s - loss: 0.2723 - accuracy: 1.0000
Epoch 69: ReduceLROnPlateau reducing learning rate to 0.00012500000593718141.
16/16 [==============================] - 6s 391ms/step - loss: 0.2723 - accuracy: 1.0000 - val_loss: 0.2997 - val_accuracy: 0.9833 - lr: 2.5000e-04
Epoch 70/100
16/16 [==============================] - 7s 414ms/step - loss: 0.2661 - accuracy: 1.0000 - val_loss: 0.2942 - val_accuracy: 0.9833 - lr: 1.2500e-04
Epoch 71/100
16/16 [==============================] - 7s 400ms/step - loss: 0.2763 - accuracy: 0.9958 - val_loss: 0.2935 - val_accuracy: 0.9833 - lr: 1.2500e-04
Epoch 72/100
16/16 [==============================] - 7s 412ms/step - loss: 0.2734 - accuracy: 0.9979 - val_loss: 0.2931 - val_accuracy: 0.9833 - lr: 1.2500e-04
Epoch 73/100
16/16 [==============================] - 7s 398ms/step - loss: 0.2683 - accuracy: 1.0000 - val_loss: 0.2894 - val_accuracy: 0.9833 - lr: 1.2500e-04
Epoch 74/100
16/16 [==============================] - 7s 399ms/step - loss: 0.2632 - accuracy: 1.0000 - val_loss: 0.2833 - val_accuracy: 0.9833 - lr: 1.2500e-04
Epoch 75/100
16/16 [==============================] - ETA: 0s - loss: 0.2637 - accuracy: 0.9979
Epoch 75: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
16/16 [==============================] - 7s 413ms/step - loss: 0.2637 - accuracy: 0.9979 - val_loss: 0.2833 - val_accuracy: 0.9833 - lr: 1.2500e-04
Epoch 76/100
16/16 [==============================] - 6s 398ms/step - loss: 0.2707 - accuracy: 0.9937 - val_loss: 0.2810 - val_accuracy: 0.9833 - lr: 6.2500e-05
Epoch 77/100
16/16 [==============================] - ETA: 0s - loss: 0.2708 - accuracy: 0.9958
Epoch 77: ReduceLROnPlateau reducing learning rate to 3.1250001484295535e-05.
16/16 [==============================] - 7s 405ms/step - loss: 0.2708 - accuracy: 0.9958 - val_loss: 0.2810 - val_accuracy: 0.9833 - lr: 6.2500e-05
Epoch 78/100
16/16 [==============================] - ETA: 0s - loss: 0.2658 - accuracy: 0.9979
Epoch 78: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
16/16 [==============================] - 7s 400ms/step - loss: 0.2658 - accuracy: 0.9979 - val_loss: 0.2811 - val_accuracy: 0.9833 - lr: 3.1250e-05
Epoch 79/100
16/16 [==============================] - ETA: 0s - loss: 0.2603 - accuracy: 1.0000
Epoch 79: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
16/16 [==============================] - 7s 403ms/step - loss: 0.2603 - accuracy: 1.0000 - val_loss: 0.2812 - val_accuracy: 0.9833 - lr: 1.5625e-05
Epoch 80/100
16/16 [==============================] - ETA: 0s - loss: 0.2594 - accuracy: 1.0000
Epoch 80: ReduceLROnPlateau reducing learning rate to 3.9062501855369196e-06.
Restoring model weights from the end of the best epoch: 77.
16/16 [==============================] - 7s 412ms/step - loss: 0.2594 - accuracy: 1.0000 - val_loss: 0.2815 - val_accuracy: 0.9833 - lr: 7.8125e-06
Epoch 80: early stopping
```

```
acc=model.evaluate( test_gen, batch_size=test_batch_size, verbose=1, steps=test_steps
msg=f'accuracy on the test set is {acc:5.2f} %'
model.save('/content/drive/MyDrive/eff_model.h5')
```

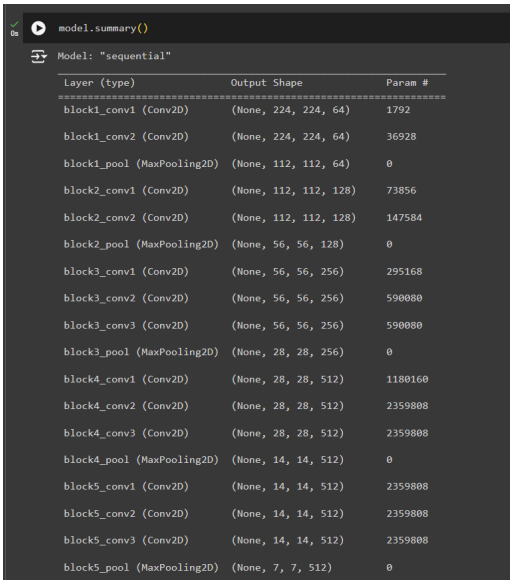
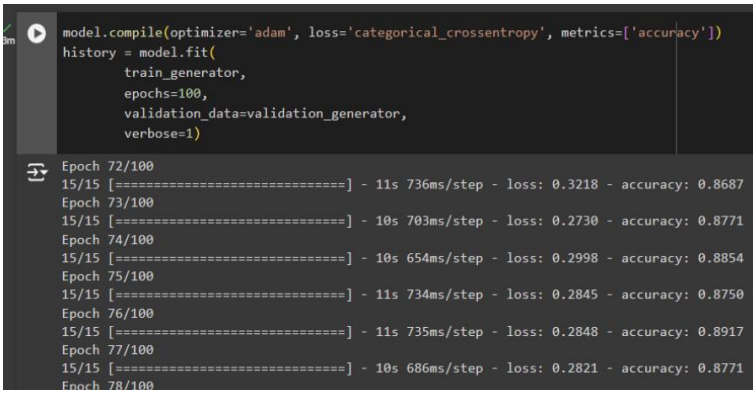
```
1/1 [==============================] - 3s 3s/step - loss: 0.3210 - accuracy: 0.9667
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0         []

 rescaling (Rescaling)       (None, 224, 224, 3)       0         ['input_2[0][0]']

 normalization (Normalizati  (None, 224, 224, 3)       7         ['rescaling[0][0]']
 on)

 rescaling_1 (Rescaling)     (None, 224, 224, 3)       0         ['normalization[0][0]']

 stem_conv_pad (ZeroPadding  (None, 225, 225, 3)       0         ['rescaling_1[0][0]']
 2D)

 stem_conv (Conv2D)          (None, 112, 112, 32)      864       ['stem_conv_pad[0][0]']

 stem_bn (BatchNormalizatio  (None, 112, 112, 32)      128       ['stem_conv[0][0]']
 n)

 stem_activation (Activatio  (None, 112, 112, 32)      0         ['stem_bn[0][0]']
 n)

 block1a_dwconv (DepthwiseC  (None, 112, 112, 32)      288       ['stem_activation[0][0]']
 onv2D)

 block1a_bn (BatchNormaliza  (None, 112, 112, 32)      128       ['block1a_dwconv[0][0]']
 tion)

 block1a_activation (Activa  (None, 112, 112, 32)      0         ['block1a_bn[0][0]']
 tion)

 block1a_se_squeeze (Global  (None, 32)                0         ['block1a_activation[0][0]']
 AveragePooling2D)

 block1a_se_reshape (Reshap  (None, 1, 1, 32)          0         ['block1a_se_squeeze[0][0]']
 e)

 block1a_se_reduce (Conv2D)  (None, 1, 1, 8)           264       ['block1a_se_reshape[0][0]']

 block1a_se_expand (Conv2D)  (None, 1, 1, 32)          288       ['block1a_se_reduce[0][0]']

 block1a_se_excite (Multipl  (None, 112, 112, 32)      0         ['block1a_activation[0][0]',
 y)                                                               'block1a_se_expand[0][0]']

 block1a_project_conv (Conv  (None, 112, 112, 16)      512       ['block1a_se_excite[0][0]']
 2D)

 block1a_project_bn (BatchN  (None, 112, 112, 16)      64        ['block1a_project_conv[0][0]']
 ormalization)
```

```
3s
block7b_bn (BatchNormaliza  (None, 7, 7, 2112)          8448      ['block7b_dwconv[0][0]']
tion)

block7b_activation (Activa  (None, 7, 7, 2112)          0         ['block7b_bn[0][0]']
tion)

block7b_se_squeeze (Global  (None, 2112)                0         ['block7b_activation[0][0]']
AveragePooling2D)

block7b_se_reshape (Reshap  (None, 1, 1, 2112)          0         ['block7b_se_squeeze[0][0]']
e)

block7b_se_reduce (Conv2D)  (None, 1, 1, 88)            185944    ['block7b_se_reshape[0][0]']

block7b_se_expand (Conv2D)  (None, 1, 1, 2112)          187968    ['block7b_se_reduce[0][0]']

block7b_se_excite (Multipl  (None, 7, 7, 2112)          0         ['block7b_activation[0][0]',
y)                                                                 'block7b_se_expand[0][0]']

block7b_project_conv (Conv  (None, 7, 7, 352)           743424    ['block7b_se_excite[0][0]']
2D)

block7b_project_bn (BatchN  (None, 7, 7, 352)           1408      ['block7b_project_conv[0][0]']
ormalization)

block7b_drop (Dropout)      (None, 7, 7, 352)           0         ['block7b_project_bn[0][0]']

block7b_add (Add)           (None, 7, 7, 352)           0         ['block7b_drop[0][0]',
                                                                   'block7a_project_bn[0][0]']

top_conv (Conv2D)           (None, 7, 7, 1408)          495616    ['block7b_add[0][0]']

top_bn (BatchNormalization  (None, 7, 7, 1408)          5632      ['top_conv[0][0]']
)

top_activation (Activation  (None, 7, 7, 1408)          0         ['top_bn[0][0]']
)

max_pool (GlobalMaxPooling  (None, 1408)                0         ['top_activation[0][0]']
2D)

batch_normalization_2 (Bat  (None, 1408)                5632      ['max_pool[0][0]']
chNormalization)

dense_3 (Dense)             (None, 256)                 360704    ['batch_normalization_2[0][0]'
                                                                   ]

dropout_2 (Dropout)         (None, 256)                 0         ['dense_3[0][0]']

dense_4 (Dense)             (None, 3)                   771       ['dropout_2[0][0]']

==================================================================================================
Total params: 8135676 (31.04 MB)
Trainable params: 8065285 (30.77 MB)
Non-trainable params: 70391 (274.97 KB)
__________________________________________________________________________________________________
```

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
VGG16		
EfficientNet