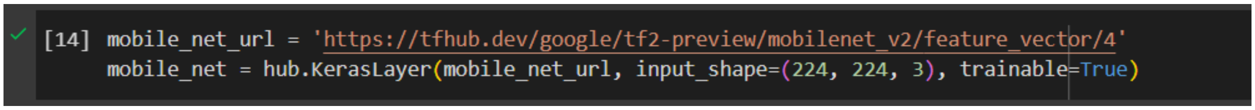
Model Development Phase

Date	14 July 2024
Team ID	SWTID1720085076
Project Title	Rice Type Classification using CNN
Maximum Marks	10 Marks

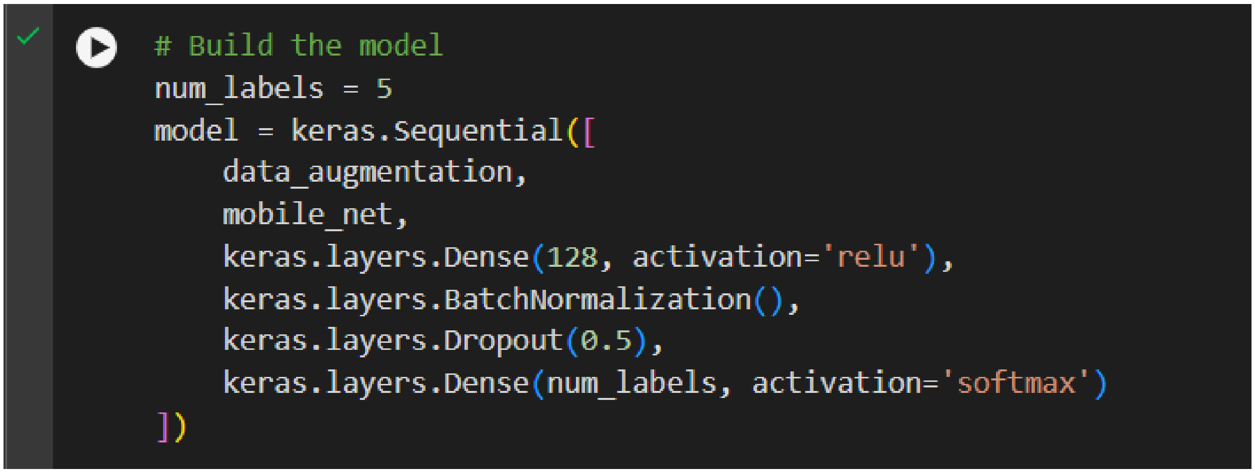
Initial Model Training Code, Model Validation and Evaluation Report

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.

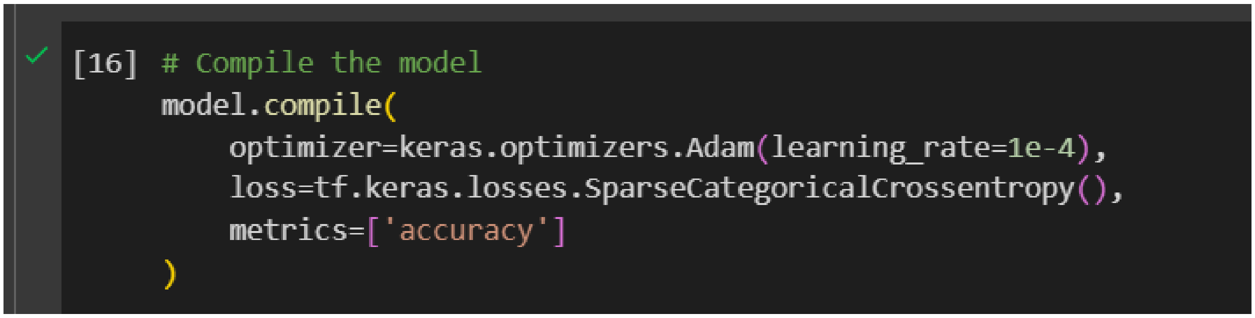
Initial Model Training Code (5 marks):



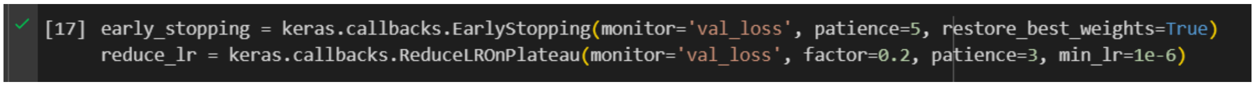
```
[14] mobile_net_url = 'https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4'
     mobile_net = hub.KerasLayer(mobile_net_url, input_shape=(224, 224, 3), trainable=True)
```



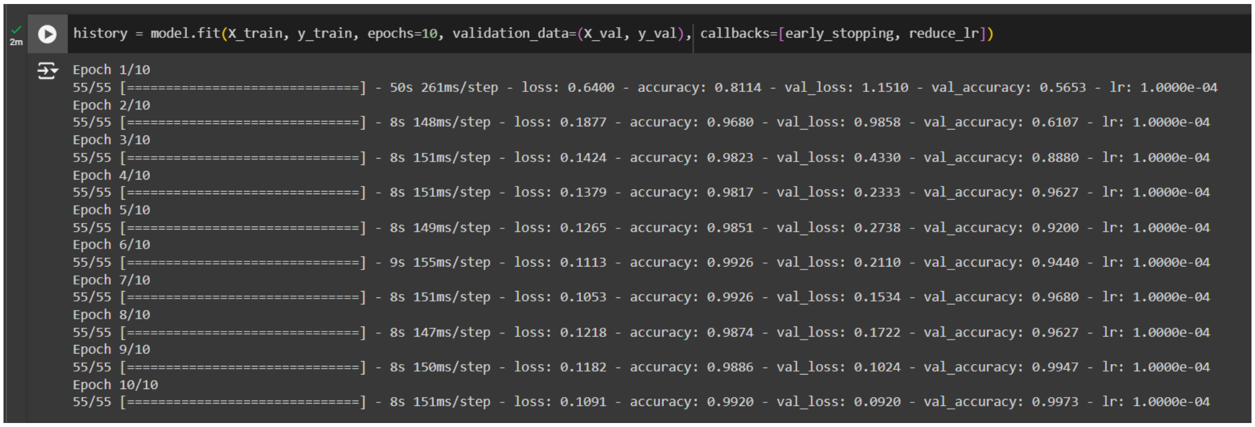
```
# Build the model
num_labels = 5
model = keras.Sequential([
    data_augmentation,
    mobile_net,
    keras.layers.Dense(128, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(num_labels, activation='softmax')
])
```



```
[16] # Compile the model
     model.compile(
         optimizer=keras.optimizers.Adam(learning_rate=1e-4),
         loss=tf.keras.losses.SparseCategoricalCrossentropy(),
         metrics=['accuracy']
     )
```

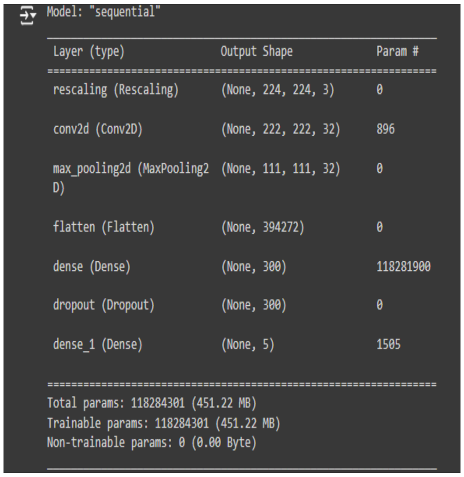
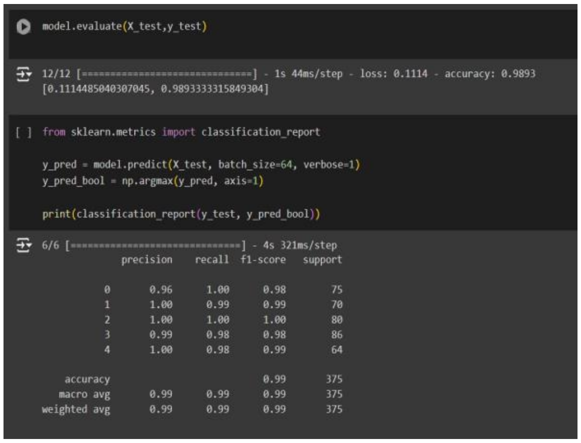


```
[17] early_stopping = keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
     reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr=1e-6)
```



```
2m  history = model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val), callbacks=[early_stopping, reduce_lr])

Epoch 1/10
55/55 [==============================] - 50s 261ms/step - loss: 0.6400 - accuracy: 0.8114 - val_loss: 1.1510 - val_accuracy: 0.5653 - lr: 1.0000e-04
Epoch 2/10
55/55 [==============================] - 8s 148ms/step - loss: 0.1877 - accuracy: 0.9680 - val_loss: 0.9858 - val_accuracy: 0.6107 - lr: 1.0000e-04
Epoch 3/10
55/55 [==============================] - 8s 151ms/step - loss: 0.1424 - accuracy: 0.9823 - val_loss: 0.4330 - val_accuracy: 0.8880 - lr: 1.0000e-04
Epoch 4/10
55/55 [==============================] - 8s 151ms/step - loss: 0.1379 - accuracy: 0.9817 - val_loss: 0.2333 - val_accuracy: 0.9627 - lr: 1.0000e-04
Epoch 5/10
55/55 [==============================] - 8s 149ms/step - loss: 0.1265 - accuracy: 0.9851 - val_loss: 0.2738 - val_accuracy: 0.9200 - lr: 1.0000e-04
Epoch 6/10
55/55 [==============================] - 9s 155ms/step - loss: 0.1113 - accuracy: 0.9926 - val_loss: 0.2110 - val_accuracy: 0.9440 - lr: 1.0000e-04
Epoch 7/10
55/55 [==============================] - 8s 151ms/step - loss: 0.1053 - accuracy: 0.9926 - val_loss: 0.1534 - val_accuracy: 0.9680 - lr: 1.0000e-04
Epoch 8/10
55/55 [==============================] - 8s 147ms/step - loss: 0.1218 - accuracy: 0.9874 - val_loss: 0.1722 - val_accuracy: 0.9627 - lr: 1.0000e-04
Epoch 9/10
55/55 [==============================] - 8s 150ms/step - loss: 0.1182 - accuracy: 0.9886 - val_loss: 0.1024 - val_accuracy: 0.9947 - lr: 1.0000e-04
Epoch 10/10
55/55 [==============================] - 8s 151ms/step - loss: 0.1091 - accuracy: 0.9920 - val_loss: 0.0920 - val_accuracy: 0.9973 - lr: 1.0000e-04
```

Model Validation and Evaluation Report (5 marks):

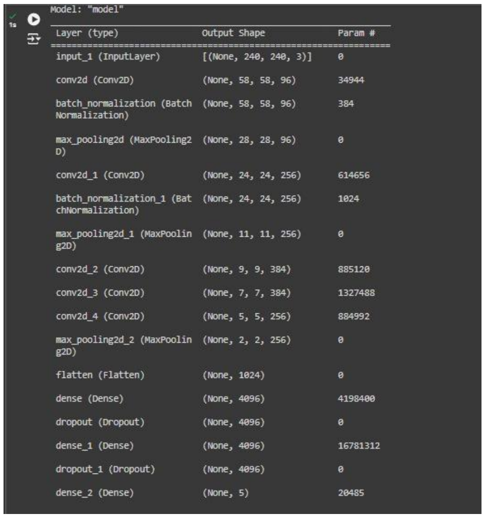
Model	Summary	Training and Validation Performance Metrics
Model 1 MobileNet		

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 224, 224, 3)       0

 conv2d (Conv2D)             (None, 222, 222, 32)      896

 max_pooling2d (MaxPooling2  (None, 111, 111, 32)      0
 D)

 flatten (Flatten)           (None, 394272)            0

 dense (Dense)               (None, 300)               118281900

 dropout (Dropout)           (None, 300)               0

 dense_1 (Dense)             (None, 5)                 1505

=================================================================
Total params: 118284301 (451.22 MB)
Trainable params: 118284301 (451.22 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

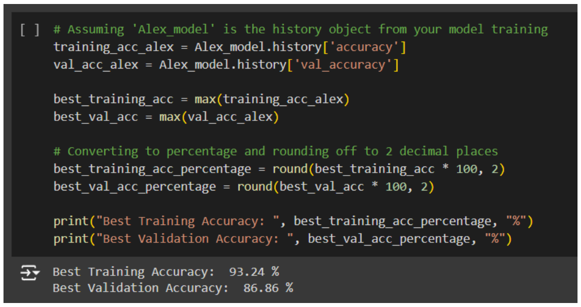
```
model.evaluate(X_test,y_test)

12/12 [==============================] - 1s 44ms/step - loss: 0.1114 - accuracy: 0.9893
[0.1114485040307045, 0.9893333315849304]

[ ] from sklearn.metrics import classification_report

    y_pred = model.predict(X_test, batch_size=64, verbose=1)
    y_pred_bool = np.argmax(y_pred, axis=1)

    print(classification_report(y_test, y_pred_bool))

6/6 [==============================] - 4s 321ms/step
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        75
           1       1.00      0.99      0.99        70
           2       1.00      1.00      1.00        80
           3       0.99      0.98      0.98        86
           4       1.00      0.98      0.99        64

    accuracy                           0.99       375
   macro avg       0.99      0.99      0.99       375
weighted avg       0.99      0.99      0.99       375
```

Model 2

ALEXNET



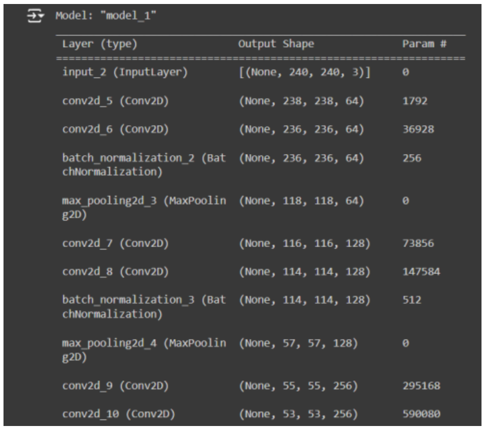
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 240, 240, 3)]     0

 conv2d (Conv2D)             (None, 58, 58, 96)        34944

 batch_normalization (Batch  (None, 58, 58, 96)        384
 Normalization)

 max_pooling2d (MaxPooling2  (None, 28, 28, 96)        0
 D)

 conv2d_1 (Conv2D)           (None, 24, 24, 256)       614656

 batch_normalization_1 (Bat  (None, 24, 24, 256)       1024
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 11, 11, 256)       0
 g2D)

 conv2d_2 (Conv2D)           (None, 9, 9, 384)         885120

 conv2d_3 (Conv2D)           (None, 7, 7, 384)         1327488

 conv2d_4 (Conv2D)           (None, 5, 5, 256)         884992

 max_pooling2d_2 (MaxPoolin  (None, 2, 2, 256)         0
 g2D)

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 4096)              4198400

 dropout (Dropout)           (None, 4096)              0

 dense_1 (Dense)             (None, 4096)              16781312

 dropout_1 (Dropout)         (None, 4096)              0

 dense_2 (Dense)             (None, 5)                 20485
```



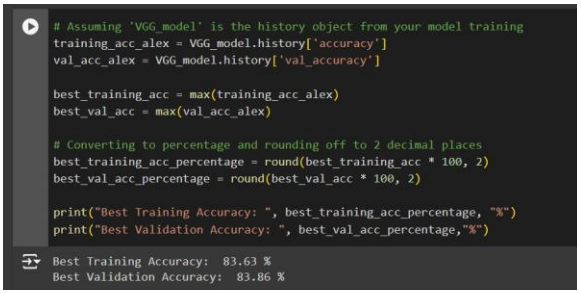
```
[ ] # Assuming 'Alex_model' is the history object from your model training
    training_acc_alex = Alex_model.history['accuracy']
    val_acc_alex = Alex_model.history['val_accuracy']

    best_training_acc = max(training_acc_alex)
    best_val_acc = max(val_acc_alex)

    # Converting to percentage and rounding off to 2 decimal places
    best_training_acc_percentage = round(best_training_acc * 100, 2)
    best_val_acc_percentage = round(best_val_acc * 100, 2)

    print("Best Training Accuracy: ", best_training_acc_percentage, "%")
    print("Best Validation Accuracy: ", best_val_acc_percentage, "%")

    Best Training Accuracy:  93.24 %
    Best Validation Accuracy:  86.86 %
```

Model 3

VGGNET



```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 240, 240, 3)]     0

 conv2d_5 (Conv2D)           (None, 238, 238, 64)      1792

 conv2d_6 (Conv2D)           (None, 236, 236, 64)      36928

 batch_normalization_2 (Bat  (None, 236, 236, 64)      256
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 118, 118, 64)      0
 g2D)

 conv2d_7 (Conv2D)           (None, 116, 116, 128)     73856

 conv2d_8 (Conv2D)           (None, 114, 114, 128)     147584

 batch_normalization_3 (Bat  (None, 114, 114, 128)     512
 chNormalization)

 max_pooling2d_4 (MaxPoolin  (None, 57, 57, 128)       0
 g2D)

 conv2d_9 (Conv2D)           (None, 55, 55, 256)       295168

 conv2d_10 (Conv2D)          (None, 53, 53, 256)       590080
```



```
# Assuming 'VGG_model' is the history object from your model training
training_acc_alex = VGG_model.history['accuracy']
val_acc_alex = VGG_model.history['val_accuracy']

best_training_acc = max(training_acc_alex)
best_val_acc = max(val_acc_alex)

# Converting to percentage and rounding off to 2 decimal places
best_training_acc_percentage = round(best_training_acc * 100, 2)
best_val_acc_percentage = round(best_val_acc * 100, 2)

print("Best Training Accuracy: ", best_training_acc_percentage, "%")
print("Best Validation Accuracy: ", best_val_acc_percentage,"%")

Best Training Accuracy:  83.63 %
Best Validation Accuracy:  83.86 %
```