Model Development Phase Template

Date	12 November 2024
Team ID	team-739690
Project Title	Tomato Plant Disease Detection From Leaf Images using DeepLearning
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):

Model Building

```
def get_model():
    base_model = ResNet152V2(input_shape=(256,256,3), include_top=False)
    for layers in base_model.layers[:140]:
        layers.trainable = False
    for layers in base_model.layers [140:]:
        layers.trainable = True
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense (1000, activation='relu')(x)
    pred=Dense(10, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=pred)
    return model
```

```
model=get_model()
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet152v2_weights_tf_dim_ordering_tf_kernels_notop.h5
234545216/234545216 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Model: "functional"
```

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 256, 256, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 262, 262, 3)	0	input_layer[0][0]
conv1_conv (Conv2D)	(None, 128, 128, 64)	9,472	conv1_pad[0][0]
pool1_pad (ZeroPadding2D)	(None, 130, 130, 64)	0	conv1_conv[0][0]
pool1_pool (MaxPooling2D)	(None, 64, 64, 64)	0	pool1_pad[0][0]
conv2_block1_preact_bn (BatchNormalization)	(None, 64, 64, 64)	256	pool1_pool[0][0]
conv2_block1_preact_relu (Activation)	(None, 64, 64, 64)	0	conv2_block1_preact_b…
conv2_block1_1_conv (Conv2D)	(None, 64, 64, 64)	4,096	conv2_block1_preact_r…
conv2_block1_1_bn (BatchNormalization)	(None, 64, 64, 64)	256	conv2_block1_1_conv[0…
conv2_block1_1_relu (Activation)	(None, 64, 64, 64)	0	conv2_block1_1_bn[0][…
conv2_block1_2_pad (ZeroPadding2D)	(None, 66, 66, 64)	0	conv2_block1_1_relu[0…

conv5_block3_2_pad (ZeroPadding2D)	(None, 10, 10, 512)	0	conv5_block3_1_relu[0…
conv5_block3_2_conv (Conv2D)	(None, 8, 8, 512)	2,359,296	conv5_block3_2_pad[0]…
conv5_block3_2_bn (BatchNormalization)	(None, 8, 8, 512)	2,048	conv5_block3_2_conv[0…
conv5_block3_2_relu (Activation)	(None, 8, 8, 512)	0	conv5_block3_2_bn[0][…
conv5_block3_3_conv (Conv2D)	(None, 8, 8, 2048)	1,050,624	conv5_block3_2_relu[0…
conv5_block3_out (Add)	(None, 8, 8, 2048)	0	conv5_block2_out[0][0… conv5_block3_3_conv[0…
post_bn (BatchNormalization)	(None, 8, 8, 2048)	8,192	conv5_block3_out[0][0]
post_relu (Activation)	(None, 8, 8, 2048)	0	post_bn[0][0]
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0	post_relu[0][0]
dense (Dense)	(None, 1000)	2,049,000	global_average_poolin…
dense_1 (Dense)	(None, 10)	10,010	dense[0][0]

```
Total params: 60,390,658 (230.37 MB)
Trainable params: 56,430,338 (215.26 MB)
Non-trainable params: 3,960,320 (15.11 MB)
```

```
model.compile(loss='categorical_crossentropy',optimizer='sgd',metrics=['accuracy'])
```

```
model.fit(train, batch_size=80,epochs=15,validation_data=val)
```

```
Epoch 1/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 2918s 13s/step - accuracy: 0.6506 - loss: 1.1542 - val_accuracy: 0.9522 - val_loss: 0.1507
Epoch 2/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 162s 669ms/step - accuracy: 0.9753 - loss: 0.0923 - val_accuracy: 0.9619 - val_loss: 0.1145
Epoch 3/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 202s 668ms/step - accuracy: 0.9949 - loss: 0.0271 - val_accuracy: 0.9725 - val_loss: 0.0901
Epoch 4/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 664ms/step - accuracy: 0.9974 - loss: 0.0137 - val_accuracy: 0.9745 - val_loss: 0.0769
Epoch 5/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 665ms/step - accuracy: 0.9980 - loss: 0.0099 - val_accuracy: 0.9715 - val_loss: 0.0885
Epoch 6/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 168s 758ms/step - accuracy: 0.9988 - loss: 0.0068 - val_accuracy: 0.9731 - val_loss: 0.0838
Epoch 7/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 183s 672ms/step - accuracy: 0.9986 - loss: 0.0061 - val_accuracy: 0.9718 - val_loss: 0.0855
Epoch 8/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 664ms/step - accuracy: 0.9995 - loss: 0.0050 - val_accuracy: 0.9672 - val_loss: 0.0942
Epoch 9/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 664ms/step - accuracy: 0.9996 - loss: 0.0050 - val_accuracy: 0.9745 - val_loss: 0.0761
Epoch 10/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 664ms/step - accuracy: 0.9996 - loss: 0.0035 - val_accuracy: 0.9751 - val_loss: 0.0793
Epoch 11/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 203s 667ms/step - accuracy: 0.9985 - loss: 0.0042 - val_accuracy: 0.9755 - val_loss: 0.0804
Epoch 12/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 663ms/step - accuracy: 0.9997 - loss: 0.0024 - val_accuracy: 0.9678 - val_loss: 0.1198
Epoch 13/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 223s 758ms/step - accuracy: 0.9991 - loss: 0.0051 - val_accuracy: 0.9731 - val_loss: 0.0811
Epoch 14/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 148s 666ms/step - accuracy: 0.9993 - loss: 0.0029 - val_accuracy: 0.9721 - val_loss: 0.0870
Epoch 15/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 664ms/step - accuracy: 0.9990 - loss: 0.0033 - val_accuracy: 0.9738 - val_loss: 0.0819
<keras.src.callbacks.history.History at 0x7ae321514910>
```

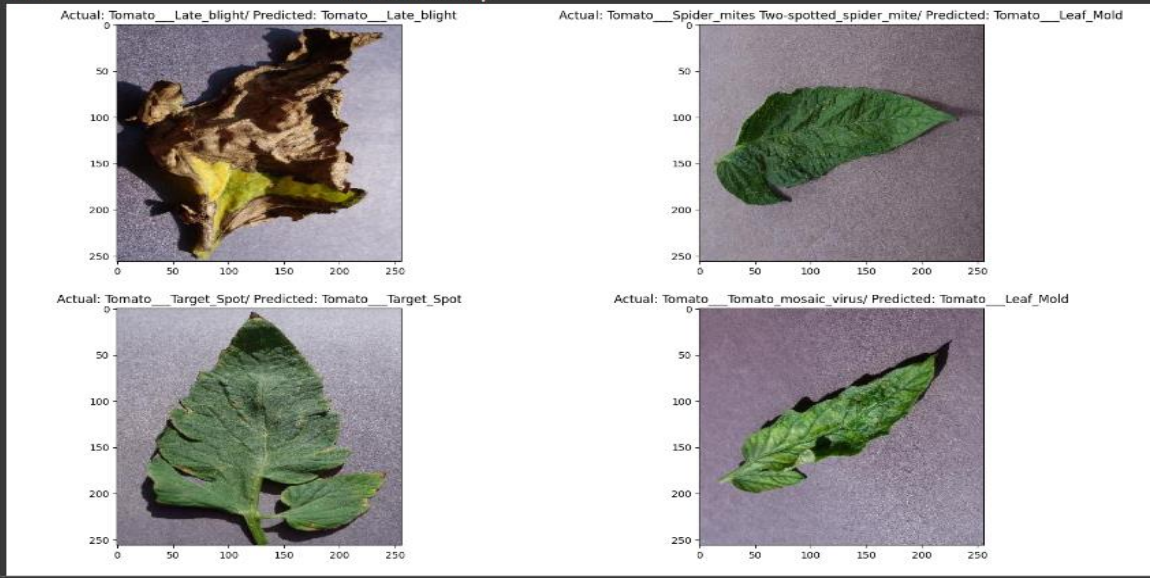
```
model.evaluate(test)
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 300s 10s/step - accuracy: 0.9616 - loss: 0.1278
[0.14101453125476837, 0.9610000252723694]
```

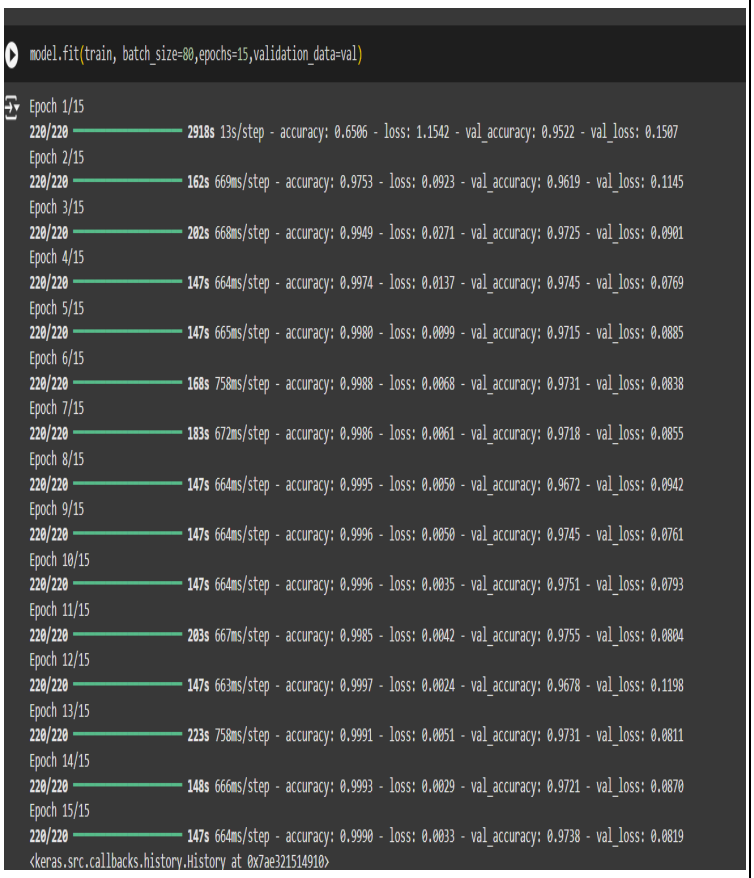
Test the Model

```
[ ]  #Prediction and visualizations

     classes = os.listdir('./val')

     plt.figure(figsize=(18,28))

     for i in enumerate(classes):

        pic = os.listdir('./val/'+i[1])

        pic = pic[np.random.randint(len(pic)-1)]

        image = Image.open('./val/' +i[1]+'/'+pic)

        image = np.asarray(image)

        pred = np.argmax(model.predict(image.reshape(-1,256,256,3)/255))
        for j in list(enumerate(list(test.class_indices.keys()))):
              if pred == j[0]:
                    prediction = j[1]
        plt.subplot(5,2,i[0]+1)

        plt.title('Actual: {0}/ Predicted: {1}'. format(i[1], prediction))

        plt.imshow(image)

     plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
```



Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
ResNet152 V2	The ResNet15v2 model achieved excellent performance for tomato plant disease detection, with a final training accuracy of 99.9% and validation accuracy stabilizing around 97.3%. The training and validation losses consistently decreased, ending at 0.0033 and 0.0819, respectively, indicating strong convergence and generalization. Each epoch processed efficiently within 147–223 seconds, showing computational efficiency. Overall, the model is highly effective and well-suited for accurate and reliable tomato plant disease detection.	

```
model.fit(train, batch_size=80,epochs=15,validation_data=val)

Epoch 1/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 2918s 13s/step - accuracy: 0.6506 - loss: 1.1542 - val_accuracy: 0.9522 - val_loss: 0.1507
Epoch 2/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 162s 669ms/step - accuracy: 0.9753 - loss: 0.0923 - val_accuracy: 0.9619 - val_loss: 0.1145
Epoch 3/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 202s 668ms/step - accuracy: 0.9949 - loss: 0.0271 - val_accuracy: 0.9725 - val_loss: 0.0901
Epoch 4/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 664ms/step - accuracy: 0.9974 - loss: 0.0137 - val_accuracy: 0.9745 - val_loss: 0.0769
Epoch 5/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 665ms/step - accuracy: 0.9980 - loss: 0.0099 - val_accuracy: 0.9715 - val_loss: 0.0885
Epoch 6/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 168s 758ms/step - accuracy: 0.9988 - loss: 0.0068 - val_accuracy: 0.9731 - val_loss: 0.0838
Epoch 7/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 183s 672ms/step - accuracy: 0.9986 - loss: 0.0061 - val_accuracy: 0.9718 - val_loss: 0.0855
Epoch 8/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 664ms/step - accuracy: 0.9995 - loss: 0.0050 - val_accuracy: 0.9672 - val_loss: 0.0942
Epoch 9/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 664ms/step - accuracy: 0.9996 - loss: 0.0050 - val_accuracy: 0.9745 - val_loss: 0.0761
Epoch 10/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 664ms/step - accuracy: 0.9996 - loss: 0.0035 - val_accuracy: 0.9751 - val_loss: 0.0793
Epoch 11/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 203s 667ms/step - accuracy: 0.9985 - loss: 0.0042 - val_accuracy: 0.9755 - val_loss: 0.0804
Epoch 12/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 663ms/step - accuracy: 0.9997 - loss: 0.0024 - val_accuracy: 0.9678 - val_loss: 0.1198
Epoch 13/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 223s 758ms/step - accuracy: 0.9991 - loss: 0.0051 - val_accuracy: 0.9731 - val_loss: 0.0811
Epoch 14/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 148s 666ms/step - accuracy: 0.9993 - loss: 0.0029 - val_accuracy: 0.9721 - val_loss: 0.0870
Epoch 15/15
220/220 ━━━━━━━━━━━━━━━━━━━━ 147s 664ms/step - accuracy: 0.9990 - loss: 0.0033 - val_accuracy: 0.9738 - val_loss: 0.0819
<keras.src.callbacks.history.History at 0x7ae321514910>
```