Model Development Phase Template

Date	15 March 2024
Team ID	SWTID1720333657
Project Title	Wce Curated Colon Disease Classification Using Deep
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):

Data Augmentation:

```
[3]: from tensorflow.keras.preprocessing.image import ImageDataGenerator

[4]: train_datagen = ImageDataGenerator(
         rotation_range=40,
         width_shift_range=0.2,
         height_shift_range=0.2,
         shear_range=0.2,
         zoom_range=0.2,
         horizontal_flip=True,
         fill_mode='nearest'
     )

[5]: train_data = train_datagen.flow_from_directory('C:/Users/DNIN/Desktop/machine learning tutorial/train',target_size = (224,224),batch_size = 32,class_mode

     Found 3200 images belonging to 4 classes.

[6]: test_data = train_datagen.flow_from_directory('C:/Users/DNIN/Desktop/machine learning tutorial/test',target_size = (224,224),batch_size = 32,class_mode =

     Found 800 images belonging to 4 classes.
```

VGG16 Model:

[1]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense
from tensorflow.keras.activations import softmax
from keras.api import activations
```

[2]:
```
Image_size = [224,224]
sol = VGG16(input_shape = Image_size + [3] ,include_top = False)
for i in sol.layers:
    i.trainable = False
y = Flatten()(sol.output)
```

[3]:
```
final = Dense(4, activation = 'softmax')(y)
vgg16_model = Model(inputs=sol.input, outputs=final)
```

11]:
```
vgg16_model.summary()
```

```
# Assuming L2 weight of 0.01
loss_weight = 0.05

vgg16_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'], loss_weights=loss_weight)
```

```
from tensorflow.keras.callbacks import EarlyStopping
# Set up early stopping to monitor validation accuracy
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3, mode = 'max')
vgg16_model.fit(train_data, epochs = 15, validation_data = test_data,  callbacks = [early_stopping])
```

[38]:
```
vgg16_model.save('cnn.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

[52]:
```
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
```

[53]:
```
labels = ['0_normal','1_ulcerative_colitis','2_polyps','3_esophagitis']
img_path = 'C:/Users/DNIN/Desktop/machine learning tutorial/test/1_ulcerative_colitis/test_ulcer_ (164).jpg'
#"C:\Users\DNIN\Desktop\machine learning tutorial\test\1_ulcerative_colitis\test_ulcer_ (164).jpg"
img = load_img(img_path, target_size=(224,224))
x = img_to_array(img)
x = preprocess_input(x)
preds = vgg16_model.predict(np.array([x]))
preds
```

ResNet-50 Model:

[4]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Flatten
from tensorflow.keras.models import Model
```

[6]:
```
Image_size = [224, 224]

# Load the pre-trained ResNet-50 model
resnet50 = ResNet50(input_shape=Image_size + [3], include_top=False)

# Freeze all layers to prevent training
for layer in resnet50.layers:
    layer.trainable = False

# Flatten the output from ResNet-50
y = tf.keras.layers.Flatten()(resnet50.output)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 17s 0us/step
```

[16]:
```
# Assuming L2 weight of 0.01
loss_weight = 0.01

resnet50_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'], loss_weights=loss_weight)
```

[21]:
```
from tensorflow.keras.callbacks import EarlyStopping

# Set up early stopping to monitor validation accuracy
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3)
# Train the model with early stopping
resnet50_model.fit(train_data, epochs=20, validation_data=test_data, callbacks=[early_stopping])
```

EfficientNet Model:

[7]:
```
from tensorflow.keras.applications import EfficientNetB0
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Flatten
from tensorflow.keras.models import Model
```

[8]:
```
Image_size = [224, 224]

efficientnet_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=Image_size + [3])
for layer in efficientnet_model.layers:
  layer.trainable = False
```

```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5
16705208/16705208 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
```

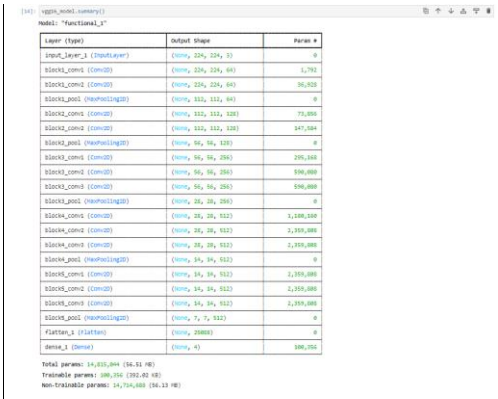
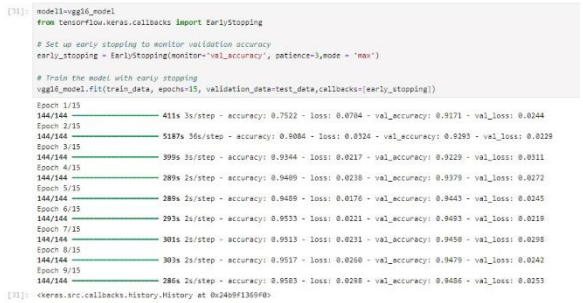
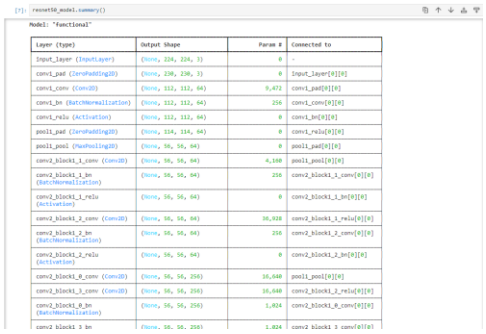
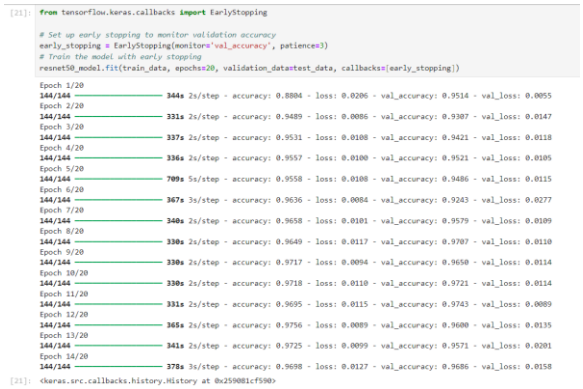
[7]:
```
from tensorflow.keras.layers import Dense
from tensorflow.keras.activations import softmax
from keras.api import activations
y = tf.keras.layers.Flatten()(efficientnet_model.output)

num_classes = 4  # Adjust based on your dataset

final_layer = Dense(num_classes, activation='softmax')(y)

efficientnet_model = Model(inputs=efficientnet_model.input, outputs=final_layer)
```

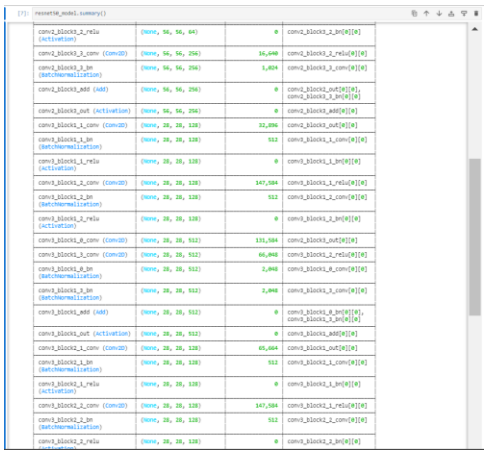
[10]:
```
efficientnet_model.summary()
```

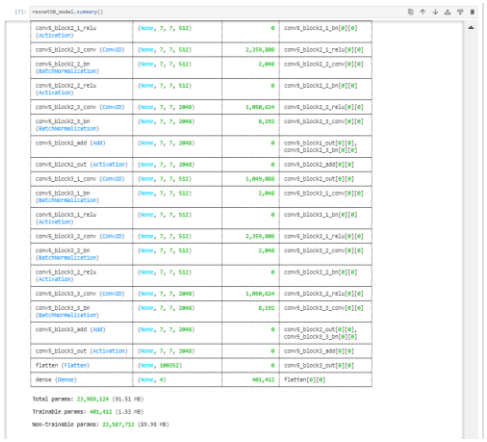
```
loss_weight = 0.05
efficientnet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'],loss_weights = loss_weight)
```

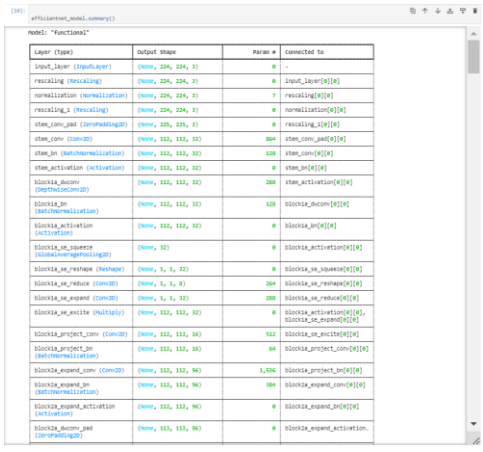
```
from tensorflow.keras.callbacks import EarlyStopping

# Set up early stopping to monitor validation accuracy
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3,mode = 'max')

# Train the model with early stopping
efficientnet_model.fit(train_data, epochs=20, validation_data = test_data , callbacks=[early_stopping])
```

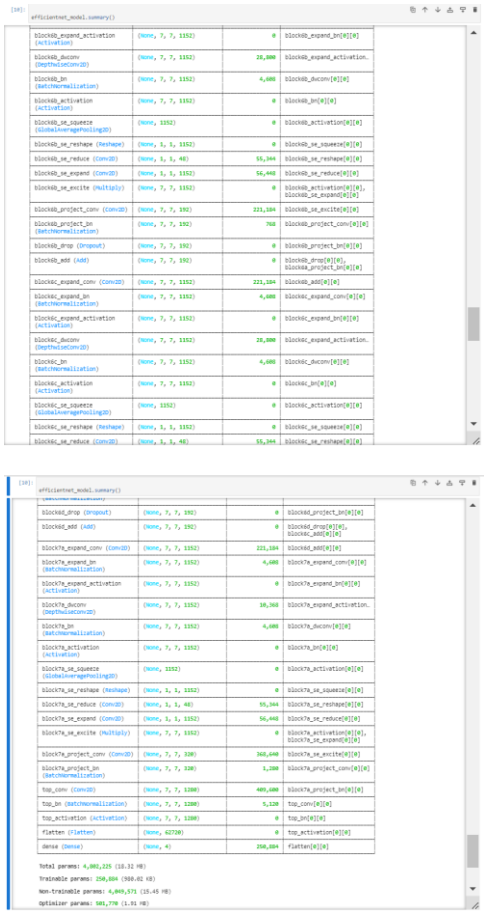
Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
VGG16 Model		
Resnet-50		





Efficient Net		



```
[10]: efficientnet_model.summary()
```

block6b_expand_activation (Activation)	(None, 7, 7, 1152)	0	block6b_expand_bn[0][0]
block6b_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	28,800	block6b_expand_activation…
block6b_bn (BatchNormalization)	(None, 7, 7, 1152)	4,608	block6b_dwconv[0][0]
block6b_activation (Activation)	(None, 7, 7, 1152)	0	block6b_bn[0][0]
block6b_se_squeeze (GlobalAveragePooling2D)	(None, 1152)	0	block6b_activation[0][0]
block6b_se_reshape (Reshape)	(None, 1, 1, 1152)	0	block6b_se_squeeze[0][0]
block6b_se_reduce (Conv2D)	(None, 1, 1, 48)	55,344	block6b_se_reshape[0][0]
block6b_se_expand (Conv2D)	(None, 1, 1, 1152)	56,448	block6b_se_reduce[0][0]
block6b_se_excite (Multiply)	(None, 7, 7, 1152)	0	block6b_activation[0][0], block6b_se_expand[0][0]
block6b_project_conv (Conv2D)	(None, 7, 7, 192)	221,184	block6b_se_excite[0][0]
block6b_project_bn (BatchNormalization)	(None, 7, 7, 192)	768	block6b_project_conv[0][0]
block6b_drop (Dropout)	(None, 7, 7, 192)	0	block6b_project_bn[0][0]
block6b_add (Add)	(None, 7, 7, 192)	0	block6b_drop[0][0], block6a_project_bn[0][0]
block6c_expand_conv (Conv2D)	(None, 7, 7, 1152)	221,184	block6b_add[0][0]
block6c_expand_bn (BatchNormalization)	(None, 7, 7, 1152)	4,608	block6c_expand_conv[0][0]
block6c_expand_activation (Activation)	(None, 7, 7, 1152)	0	block6c_expand_bn[0][0]
block6c_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	28,800	block6c_expand_activation…
block6c_bn (BatchNormalization)	(None, 7, 7, 1152)	4,608	block6c_dwconv[0][0]
block6c_activation (Activation)	(None, 7, 7, 1152)	0	block6c_bn[0][0]
block6c_se_squeeze (GlobalAveragePooling2D)	(None, 1152)	0	block6c_activation[0][0]
block6c_se_reshape (Reshape)	(None, 1, 1, 1152)	0	block6c_se_squeeze[0][0]
block6c_se_reduce (Conv2D)	(None, 1, 1, 48)	55,344	block6c_se_reshape[0][0]


```
[10]: efficientnet_model.summary()
```

block6d_drop (Dropout)	(None, 7, 7, 192)	0	block6d_project_bn[0][0]
block6d_add (Add)	(None, 7, 7, 192)	0	block6d_drop[0][0], block6c_add[0][0]
block7a_expand_conv (Conv2D)	(None, 7, 7, 1152)	221,184	block6d_add[0][0]
block7a_expand_bn (BatchNormalization)	(None, 7, 7, 1152)	4,608	block7a_expand_conv[0][0]
block7a_expand_activation (Activation)	(None, 7, 7, 1152)	0	block7a_expand_bn[0][0]
block7a_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	10,368	block7a_expand_activation…
block7a_bn (BatchNormalization)	(None, 7, 7, 1152)	4,608	block7a_dwconv[0][0]
block7a_activation (Activation)	(None, 7, 7, 1152)	0	block7a_bn[0][0]
block7a_se_squeeze (GlobalAveragePooling2D)	(None, 1152)	0	block7a_activation[0][0]
block7a_se_reshape (Reshape)	(None, 1, 1, 1152)	0	block7a_se_squeeze[0][0]
block7a_se_reduce (Conv2D)	(None, 1, 1, 48)	55,344	block7a_se_reshape[0][0]
block7a_se_expand (Conv2D)	(None, 1, 1, 1152)	56,448	block7a_se_reduce[0][0]
block7a_se_excite (Multiply)	(None, 7, 7, 1152)	0	block7a_activation[0][0], block7a_se_expand[0][0]
block7a_project_conv (Conv2D)	(None, 7, 7, 320)	368,640	block7a_se_excite[0][0]
block7a_project_bn (BatchNormalization)	(None, 7, 7, 320)	1,280	block7a_project_conv[0][0]
top_conv (Conv2D)	(None, 7, 7, 1280)	409,600	block7a_project_bn[0][0]
top_bn (BatchNormalization)	(None, 7, 7, 1280)	5,120	top_conv[0][0]
top_activation (Activation)	(None, 7, 7, 1280)	0	top_bn[0][0]
flatten (Flatten)	(None, 62720)	0	top_activation[0][0]
dense (Dense)	(None, 4)	250,884	flatten[0][0]

```
Total params: 4,802,225 (18.32 MB)
Trainable params: 250,884 (980.02 KB)
Non-trainable params: 4,049,571 (15.45 MB)
Optimizer params: 501,770 (1.91 MB)
```