Model Development Phase

Date	09 - JULY- 2024
Team ID	SWTID1719999219
Project Title	Crystal Clear Vision: Revolutionizing Cataract Prediction through Transfer Learning Mastery
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):

Paste the screenshot of the model training code

RESNET50 -

Resnet50

First, we will do feature extraction with keras functional API

```
[ ]  #We are going to use EarlyStopping. Basically what it does is, it monitors validation accuracy.
     #If validation accuracy does not improve for 3 epochs, the model fitting will cease.
     from keras.callbacks import EarlyStopping, ModelCheckpoint
```

```
#resnet 50
base_model= tf.keras.applications.ResNet50(include_top=False)
#include top is false as we are going to create our own output layers for the model

base_model.trainable=False
#freezing the base layers

#input layer
inputs=tf.keras.layers.Input(shape=(224,224,3),name='input layer')

#adding in data augmentation as a layer itself
x=data_augmentation(inputs)

#pass inputs to base model
x=base_model(x, training=False)
print(x.shape)

#We will use average pooling to reduce the size of the feature map.
x=tf.keras.layers.GlobalAveragePooling2D(name='global_average_pooling_layer')(x)
print(x.shape)

outputs=tf.keras.layers.Dense(1,activation='sigmoid',name='output_layer')(x)
model_0= tf.keras.Model(inputs,outputs)

#define early stopping callback.
early_stopping=EarlyStopping(monitor='val_accuracy',patience=3,restore_best_weights=True)
checkpoint = ModelCheckpoint('best_resnet_model.h5', monitor='val_accuracy', save_best_only=True)

model_0.compile(loss='binary_crossentropy',
                optimizer=tf.keras.optimizers.Adam(),
                metrics=['accuracy'])

#Using 50 epochs instead of 5
history_model_0= model_0.fit(train_data,
                             epochs=50,
                             validation_data=val_data,
                             callbacks=[early_stopping,checkpoint])
```

VGG16 –

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

# Load VGG16 base model
vgg = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze VGG16 layers
for layer in vgg.layers:
    layer.trainable = False

# Create your model on top of VGG16
x = Flatten()(vgg.output)
output = Dense(1, activation='sigmoid')(x)
model = Model(vgg.input, output)

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
train_path = os.path.join(base_path, "train")
validation_path = os.path.join(base_path, "validation")
model_save_path = os.path.join(base_path, "model/vgg16_model.h5")

train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
validation_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_path, target_size=(224, 224), batch_size=16, class_mode='binary')
validation_generator = validation_datagen.flow_from_directory(validation_path, target_size=(224, 224), batch_size=16, class_mode='binary')

##model.fit(train_generator, validation_data=validation_generator, epochs=2)
# Define callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
checkpoint = ModelCheckpoint(model_save_path, monitor='val_loss', save_best_only=True, verbose=1)

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])

# Train the model with callbacks
history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=50,
    callbacks=[early_stopping, checkpoint]
)

# Print the model save path
print(f"Model saved at: {model_save_path}")
```

Efficient B1 –

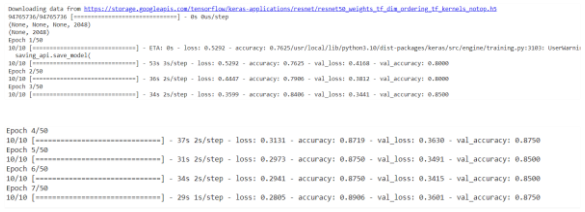
```
base_model= tf.keras.applications.EfficientNetB1(include_top=False)
#include top is false as we are going to create our own output layers for the model

base_model.trainable=False
#freezing the base layers

#input layer
inputs=tf.keras.layers.Input(shape=(240,240,3),name='input layer')


#adding in data augmentation as a layer itself
x=data_augmentation(inputs)

#pass inputs to base model
x=base_model(x, training=False)
print(x.shape)

#We will use average pooling to reduce the size of the feature map.
x=tf.keras.layers.GlobalAveragePooling2D(name='global_average_pooling_layer')(x)
print(x.shape)

outputs=tf.keras.layers.Dense(1,activation='sigmoid',name='output_layer')(x)
model_0= tf.keras.Model(inputs,outputs)
```

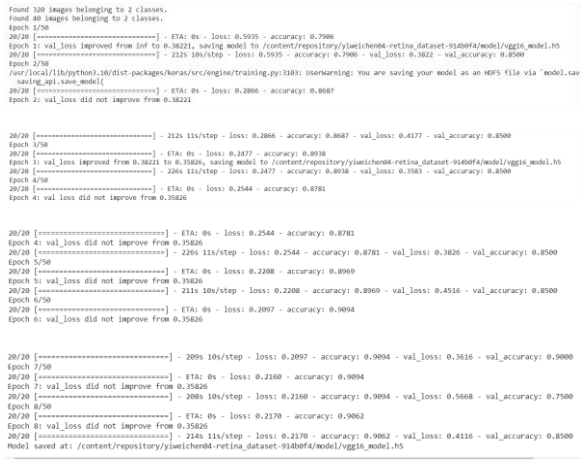
```
#define early stopping callback.
early_stopping=EarlyStopping(monitor='val_accuracy',patience=3,restore_best_weights=True)
checkpoint = ModelCheckpoint('best_efficientnet_model.h5', monitor='val_accuracy', save_best_only=True)

model_0.compile(loss='binary_crossentropy',
                optimizer=tf.keras.optimizers.Adam(),
                metrics=['accuracy'])

history_model_0= model_0.fit(train_data,
                             epochs=50,
                             validation_data=val_data,
                             callbacks=[early_stopping,checkpoint])
```

InceptionV3 Model –

```
# InceptionV3 model
inception = InceptionV3(include_top=False, input_shape=(299, 299, 3))
x = Flatten()(inception.output)
output = Dense(2, activation='softmax')(x)  # Changed to 2 classes for binary classification

inception_model = Model(inception.input, output)
inception_model.summary()

# Compile the model
inception_model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])

# Define Early Stopping and Model Checkpoint callbacks
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3, restore_best_weights=True)
checkpoint = ModelCheckpoint('best_inception_model.h5', monitor='val_accuracy', save_best_only=True)

# Train the model with early stopping and model checkpoint callbacks
history = inception_model.fit(train_gen, validation_data=val_gen, epochs=epochs, callbacks=[early_stopping, checkpoint])
```

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
Resnet50	`model_0.summary()` Model: "model" _____ Layer (type) Output Shape Param # ===== input layer (InputLayer) [(None, 224, 224, 3)] 0 data_augmentation (Sequential) (None, None, None, 3) 0 resnet50 (Functional) (None, None, None, 2048) 23587712 global_average_pooling_layer (GlobalAveragePooling2D) (None, 2048) 0 output_layer (Dense) (None, 1) 2049 ===== Total params: 23589761 (89.99 MB) Trainable params: 2049 (8.00 KB) Non-trainable params: 23587712 (89.98 MB)	

VGG16	`# Print the model summary`
	`model.summary()`
	
	Downloading data from https://storage.googleapis.com/tensorflow/
	58889256/58889256 [==========] - 0s 0us/step
	Model: "model"
	_____
	Layer (type) Output Shape Param #
	=====
	input_1 (InputLayer) [(None, 224, 224, 3)] 0
	
	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, 1) 25089 Total params: 14739777 (56.23 MB) Trainable params: 25089 (98.00 KB) Non-trainable params: 14714688 (56.13 MB)	
EfficientnetB1	model_0.summary() Model: "model_1" Layer (type) Output Shape Param # input layer (InputLayer) [(None, 240, 240, 3)] 0 data_augmentation (Sequential) (None, None, None, 3) 0 efficientnetb1 (Functional) (None, None, None, 1280) 6575239 global_average_pooling_layer (GlobalAveragePooling2D) (None, 1280) 0 output_layer (Dense) (None, 1) 1281 Total params: 6576520 (25.09 MB) Trainable params: 1281 (5.00 KB) Non-trainable params: 6575239 (25.08 MB)	Epoch 1/50 10/10 [==========] - ETA: 0s - loss: 0.5833 - accuracy: 0.7031/usr/local/lib/python3.10/dist-packages/keras/src saving_api.save_model(10/10 [==========] - 49s 2s/step - loss: 0.5833 - accuracy: 0.7031 - val_loss: 0.5492 - val_accuracy: 0.7500 Epoch 2/50 10/10 [==========] - 29s 1s/step - loss: 0.4924 - accuracy: 0.7594 - val_loss: 0.4763 - val_accuracy: 0.7500 Epoch 3/50 10/10 [==========] - 31s 1s/step - loss: 0.4256 - accuracy: 0.8094 - val_loss: 0.4104 - val_accuracy: 0.8000 Epoch 4/50 10/10 [==========] - 28s 1s/step - loss: 0.3824 - accuracy: 0.8562 - val_loss: 0.4005 - val_accuracy: 0.8000 Epoch 5/50 10/10 [==========] - 27s 1s/step - loss: 0.3291 - accuracy: 0.8594 - val_loss: 0.3827 - val_accuracy: 0.8000 Epoch 6/50 10/10 [==========] - 28s 1s/step - loss: 0.3217 - accuracy: 0.8719 - val_loss: 0.3631 - val_accuracy: 0.8250 Epoch 7/50 10/10 [==========] - 27s 930ms/step - loss: 0.3006 - accuracy: 0.8813 - val_loss: 0.3517 - val_accuracy: 0.8250 Epoch 8/50 10/10 [==========] - 26s 1s/step - loss: 0.2921 - accuracy: 0.8906 - val_loss: 0.3606 - val_accuracy: 0.8250 Epoch 9/50 10/10 [==========] - 26s 1s/step - loss: 0.2749 - accuracy: 0.8875 - val_loss: 0.3477 - val_accuracy: 0.8250
Inception	activation_186 (Activation) (None, 8, 8, 384) 0 ['batch_normalization_186[0][0]'] batch_normalization_187 (BatchNormalization) (None, 8, 8, 192) 576 ['conv2d_187[0][0]'] activation_179 (Activation) (None, 8, 8, 320) 0 ['batch_normalization_179[0][0]'] mixed9_1 (Concatenate) (None, 8, 8, 768) 0 ['activation_181[0][0]', 'activation_182[0][0]'] concatenate_3 (Concatenate) (None, 8, 8, 768) 0 ['activation_185[0][0]', 'activation_186[0][0]'] activation_187 (Activation) (None, 8, 8, 192) 0 ['batch_normalization_187[0][0]'] mixed10 (Concatenate) (None, 8, 8, 2048) 0 ['activation_179[0][0]', 'mixed9_1[0][0]', 'concatenate_3[0][0]', 'activation_187[0][0]'] flatten_1 (Flatten) (None, 131072) 0 ['mixed10[0][0]'] dense_1 (Dense) (None, 2) 262146 ['flatten_1[0][0]'] Total params: 22062498 (84.17 MB) Trainable params: 22030498 (84.04 MB) Non-trainable params: 34432 (134.50 KB)	Epoch 1/50 10/10 [==========] - ETA: 0s - loss: 5.6772 - accuracy: 0.7219 /usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: saving_api.save_model(10/10 [==========] - 386s 36s/step - loss: 5.6772 - accuracy: 0.7219 - val_loss: 821.4874 - val_accuracy: 0.7500 Epoch 2/50 10/10 [==========] - 411s 41s/step - loss: 1.6518 - accuracy: 0.8281 - val_loss: 152897.5312 - val_accuracy: 0.7500 Epoch 3/50 10/10 [==========] - 420s 41s/step - loss: 0.4699 - accuracy: 0.8500 - val_loss: 19500656.0000 - val_accuracy: 0.7500 Epoch 4/50 10/10 [==========] - 347s 34s/step - loss: 0.3396 - accuracy: 0.8656 - val_loss: 16894314.0000 - val_accuracy: 0.7500 Model saved at: /content/best_inception_model.h5 2/2 [==========] - 13s 2s/step - loss: 870.6385 - accuracy: 0.7500 Test accuracy: 0.75 2/2 [==========] - 14s 2s/step