```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns



#import model building libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten


from keras.preprocessing.image import ImageDataGenerator

#2.configure image data generator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)


#3.Apply image data generator functionality to train and test images
x_train=train_datagen.flow_from_directory(r'drive/MyDrive/train/dataset',target_size=(64,64),batch_size=32,class_mode="categorical")
x_test = test_datagen.flow_from_directory(r'drive/MyDrive/test',target_size = (64,64),batch_size=32,class_mode="categorical")
```

```
Found 4217 images belonging to 4 classes.
Found 400 images belonging to 4 classes.
```

```
print(x_train.class_indices)
```

```
{'cataract': 0, 'diabetic_retinopathy': 1, 'glaucoma': 2, 'normal': 3}
```

```
model=Sequential()


model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation="relu"))


#add max pool layer(pool_size)
model.add(MaxPooling2D(pool_size=(2,2)))


#add flatten layer  ---input of ann
model.add(Flatten())


#ann hidden layer
model.add(Dense(units=128,activation="relu"))


#add output layer
model.add(Dense(units=4,activation="softmax"))


#Compile the model (loss fucntion,accuracy,optimizer)
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics="accuracy")


#fit model (x_train,steps_per epoch,epochs,validation_data,validation_steps)
model.fit(x_train,epochs=16,validation_data=x_test,validation_steps=10)
```

```
Epoch 1/16
132/132 [==============================] - 947s 7s/step - loss: 1.1645 - accuracy: 0.4885 - val_loss: 1.0293 - val_accuracy: 0.5344
Epoch 2/16
132/132 [==============================] - 78s 592ms/step - loss: 0.9044 - accuracy: 0.6182 - val_loss: 0.9396 - val_accuracy: 0.6156
Epoch 3/16
132/132 [==============================] - 67s 512ms/step - loss: 0.7976 - accuracy: 0.6602 - val_loss: 0.8152 - val_accuracy: 0.6375
Epoch 4/16
132/132 [==============================] - 67s 511ms/step - loss: 0.7355 - accuracy: 0.6908 - val_loss: 0.7618 - val_accuracy: 0.6875
Epoch 5/16
132/132 [==============================] - 68s 517ms/step - loss: 0.7024 - accuracy: 0.7161 - val_loss: 0.5992 - val_accuracy: 0.7469
Epoch 6/16
132/132 [==============================] - 67s 503ms/step - loss: 0.6695 - accuracy: 0.7235 - val_loss: 0.7361 - val_accuracy: 0.7031
Epoch 7/16
132/132 [==============================] - 67s 509ms/step - loss: 0.6481 - accuracy: 0.7392 - val_loss: 0.6237 - val_accuracy: 0.7812
Epoch 8/16
132/132 [==============================] - 67s 506ms/step - loss: 0.6535 - accuracy: 0.7356 - val_loss: 0.5331 - val_accuracy: 0.8281
Epoch 9/16
```

```
132/132 [==============================] - 66s 497ms/step - loss: 0.6064 - accuracy: 0.7574 - val_loss: 0.5779 - val_accuracy: 0.7969
Epoch 10/16
132/132 [==============================] - 74s 548ms/step - loss: 0.5918 - accuracy: 0.7636 - val_loss: 0.5969 - val_accuracy: 0.7844
Epoch 11/16
132/132 [==============================] - 66s 501ms/step - loss: 0.5616 - accuracy: 0.7771 - val_loss: 0.6128 - val_accuracy: 0.7656
Epoch 12/16
132/132 [==============================] - 67s 504ms/step - loss: 0.5506 - accuracy: 0.7778 - val_loss: 0.5530 - val_accuracy: 0.7937
Epoch 13/16
132/132 [==============================] - 67s 505ms/step - loss: 0.5383 - accuracy: 0.7828 - val_loss: 0.4341 - val_accuracy: 0.8562
Epoch 14/16
132/132 [==============================] - 67s 504ms/step - loss: 0.5318 - accuracy: 0.7882 - val_loss: 0.5111 - val_accuracy: 0.8125
Epoch 15/16
132/132 [==============================] - 68s 511ms/step - loss: 0.5088 - accuracy: 0.7939 - val_loss: 0.6380 - val_accuracy: 0.7375
Epoch 16/16
132/132 [==============================] - 66s 501ms/step - loss: 0.5098 - accuracy: 0.7970 - val_loss: 0.4866 - val_accuracy: 0.8281
<keras.callbacks.History at 0x7ed6d2944fa0>
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 flatten (Flatten)           (None, 30752)             0

 dense (Dense)               (None, 128)               3936384

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 3,937,796
Trainable params: 3,937,796
Non-trainable params: 0
_________________________________________________________________
```

```
model.save("EyeDis.h5")
```

Testing the model

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

```
import tensorflow as tf
```

```
model=tf.keras.models.load_model(r"/content/EyeDis.h5",compile=False)
```

```
#D:\SmartBridge\VIT_morning_slot\dataset\Testing\elephants\nature_3306013__340.jpg
img=image.load_img(r'drive/MyDrive/test/cataract/_1_5346540.jpg',target_size=(64,64))
```

```
img
```

```
x=image.img_to_array(img)
```

```
x
```

```
array([[[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],
```

```
       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       ...,

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]]], dtype=float32)
```

```
x=np.expand_dims(x,axis=0)
```

```
x.ndim
```

```
4
```

```
x.shape
```

```
(1, 64, 64, 3)
```

```
pred=model.predict(x)
```

```
1/1 [==============================] - 0s 93ms/step
```

```
pred
```

```
array([[1., 0., 0., 0.]], dtype=float32)
```

```
{'catarct': 0, 'diabetic_retinopathy': 1, 'glaucoma': 2, 'normal': 3}
```

```
{'catarct': 0, 'diabetic_retinopathy': 1, 'glaucoma': 2, 'normal': 3}
```

```
pred_class=np.argmax(pred,axis=1)
```

```
pred_class[0]
```

```
0
```

```
index = ['cataract','diabetic_retinopathy','glaucoma','normal']
result = str(index[pred_class[0]])
print(result)
```

```
cataract
```