Imported Libraries

In [1]:
```
import tensorflow as tf
import keras
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout, BatchNormalization, LSTM, Input, Re
shape
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.losses import sparse_categorical_crossentropy
from tensorflow.keras.optimizers import RMSprop
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
import cv2
import os
```

Image Dataset Import

In [2]:
```
labels = ['1_normal', '2_cataract','3_glaucoma','4_retina_disease']
img_size = 224
def get_data(data_dir):
    data = []

    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img))[...,::-1] #convert BGR to RGB format
                crop_image= img_arr[0:1728,430:2190]
                resized_arr = cv2.resize(crop_image, (img_size, img_size)) # Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

In [3]:
```
#function call to get_data function that takes file path of the dataset.
data= get_data('dataset/dataset_all_equal_size_image/')
```

```
<ipython-input-2-b08f5e223f84>:17: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which
is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do
this, you must specify 'dtype=object' when creating the ndarray
  return np.array(data)
```

In [4]:
```
data.shape
```

Out[4]: (600, 2)

In [5]:
```
type(data)
```

Out[5]: numpy.ndarray

Dividing Data Ndarray into Normal, Cataract, Glaucoma and Retina diseases.

In [6]:
```
normal= data[0:300]
normal.shape
```

Out[6]: (300, 2)

In [7]:
```
cataract=data[300:400]
cataract.shape
```

Out[7]: (100, 2)

In [8]:
```
glaucoma= data[400:500]
glaucoma.shape
```

Out[8]: (100, 2)

In [9]:
```
retina_disease= data[500:600]
retina_disease.shape
```

Out[9]:
```
(100, 2)
```

In [10]:
```
random.seed(10)
np.random.shuffle(normal)
np.random.shuffle(cataract)
np.random.shuffle(glaucoma)
np.random.shuffle(retina_disease)
```

Performing Normalization and Resize operation

In [11]:
```
def normalize(x_train,x_val,x_test):

    x_train = np.array(x_train) / 255
    x_train.reshape(-1, img_size, img_size, 1)

    x_test= np.array(x_test) / 255
    x_test.reshape(-1, img_size, img_size, 1)


    x_val= np.array(x_val) / 255
    x_val.reshape(-1, img_size, img_size, 1)


    return (x_train,x_val,x_test)
```

Separating the Images and Labels into Respective Variables

In [12]:
```
def image_label_split(train,validation,test):

    x_train = []
    y_train = []
    x_val = []
    y_val = []
    x_test = []
    y_test = []

    for feature, label in train:
      x_train.append(feature)
      y_train.append(label)

    for feature, label in validation:
      x_val.append(feature)
      y_val.append(label)


    for feature, label in test:
      x_test.append(feature)
      y_test.append(label)


    y_train = np.array(y_train)
    y_val = np.array(y_val)
    y_test= np.array(y_test)

    return (x_train,y_train,x_val,y_val,x_test,y_test)
```

InceptionV3-LSTM MODEL

In [13]:
```
def model_build_compile(k):
    baseModel = InceptionV3(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))
    for layer in baseModel.layers:
            layer.trainable = False

    x = baseModel.output

        # LSTM layer
    x = Reshape((25, 2048))(x)
    x = ((LSTM(512, activation="relu", return_sequences=True, trainable=False)))(x)
    x = BatchNormalization()(x)

        # FC layer
    x = Flatten(name="flatten")(x)

        # fc1 layer
    x = Dense(units=4096, activation='relu')(x)
    x = BatchNormalization()(x)


        # fc2 layer
    x = Dense(units=4096, activation='relu')(x)
    x = BatchNormalization()(x)

        # Output layer
    output = Dense(units=4, activation='softmax')(x)

    model = Model(inputs=baseModel.input, outputs=output)
    opt = RMSprop(learning_rate=0.01, clipvalue=100)
    model.compile(loss='sparse_categorical_crossentropy', optimizer=opt, metrics=["accuracy"])
    k=k+1
    print("model building and compiling for fold",k)

    return model
```

Model prediction for Test Images and Computation of Sensitivity and Specificity

In [14]:
```
def test_pred(x_val,y_val,k):
    predictions = model.predict(x_val)
    predictions = np.argmax(predictions, axis = -1)

    print('----------------Test accuracy for',k+1,'fold----------------')
    #Confusion matrix, Accuracy, sensitivity and specificity
    cm1 = confusion_matrix(y_val,predictions)
    print('Confusion Matrix : \n', cm1)

    #####from confusion matrix calculate accuracy

    sensitivity_1_normal = (cm1[0,0])/(cm1[0,0]+cm1[0,1]+cm1[0,2]+cm1[0,3])
    #print('Sensitivity_1_normal          : ', sensitivity_1_normal )

    sensitivity_2_cataract = (cm1[1,1])/(cm1[1,0]+cm1[1,1]+cm1[1,2]+cm1[1,3])
    #print('Sensitivity_2_cataract        : ', sensitivity_2_cataract )


    sensitivity_3_glaucoma = (cm1[2,2])/(cm1[2,0]+cm1[2,1]+cm1[2,2]+cm1[2,3])
    #print('Sensitivity_3_glaucoma        : ', sensitivity_3_glaucoma )

    sensitivity_4_retina_disease = (cm1[3,3])/(cm1[3,0]+cm1[3,1]+cm1[3,2]+cm1[3,3])
    #print('Sensitivity_4_retina_disease  : ', sensitivity_4_retina_disease )

    specificity_1_normal = (cm1[1,1]+cm1[1,2]+cm1[1,3]+cm1[2,1]+cm1[2,2]+cm1[2,3]+cm1[3,1]+cm1[3,2]+cm1[3,3])/(cm1[1,0
]+cm1[2,0]+cm1[3,0]+cm1[1,1]+cm1[1,2]+cm1[1,3]+cm1[2,1]+cm1[2,2]+cm1[2,3]+cm1[3,1]+cm1[3,2]+cm1[3,3])
    #print('Specificity : ', specificity_1_normal)

    specificity_2_cataract = (cm1[0,0]+cm1[0,2]+cm1[0,3]+cm1[2,0]+cm1[2,2]+cm1[2,3]+cm1[3,0]+cm1[3,2]+cm1[3,3])/(cm1[0
,1]+cm1[2,1]+cm1[3,1]+cm1[0,0]+cm1[0,2]+cm1[0,3]+cm1[2,0]+cm1[2,2]+cm1[2,3]+cm1[3,0]+cm1[3,2]+cm1[3,3])
    #print('Specificity : ', specificity_2_cataract)

    specificity_3_glaucoma = (cm1[0,0]+cm1[0,1]+cm1[0,3]+cm1[1,0]+cm1[1,1]+cm1[1,3]+cm1[3,0]+cm1[3,1]+cm1[3,3])/(cm1[0
,2]+cm1[1,2]+cm1[3,2]+cm1[0,0]+cm1[0,1]+cm1[0,3]+cm1[1,0]+cm1[1,1]+cm1[1,3]+cm1[3,0]+cm1[3,1]+cm1[3,3])
    #print('Specificity : ', specificity_3_glaucoma)

    specificity_4_retina_disease= (cm1[0,0]+cm1[0,1]+cm1[0,2]+cm1[1,0]+cm1[1,1]+cm1[1,2]+cm1[2,0]+cm1[2,1]+cm1[2,2])/(
cm1[0,3]+cm1[1,3]+cm1[2,3]+cm1[0,0]+cm1[0,1]+cm1[0,2]+cm1[1,0]+cm1[1,1]+cm1[1,2]+cm1[2,0]+cm1[2,1]+cm1[2,2])
    #print('Specificity : ', specificity_4_retina_disease)

    Sensitivity= (sensitivity_1_normal + sensitivity_2_cataract + sensitivity_3_glaucoma + sensitivity_4_retina_diseas
e)/4
    #print(Sensitivity)

    Specificity= (specificity_1_normal + specificity_2_cataract + specificity_3_glaucoma + specificity_4_retina_diseas
e)/4
    #print(Specificity)

    total1=sum(sum(cm1))
    test_accuracy=(cm1[0,0]+cm1[1,1]+cm1[2,2]+cm1[3,3])/total1

    print ('Accuracy    : ', test_accuracy)
    print ('Specificity : ', Specificity)
    print ('Sensitivity : ', Sensitivity)
    print('-----------------------------End of',k+1,'Fold-------------------------')
    return test_accuracy,Specificity,Sensitivity,cm1
```

In [15]:
```
CM= []
test_accuracy=[]
test_sensitivity=[]
test_specificity=[]
train_acc = []
val_acc = []
train_loss = []
val_loss = []
```

InceptionV3-LSTM 5 Fold Cross Validation

In [16]:
```
for k in range (5):  # for loop to run 5 folds
    n_normal=30      # specifying the number of images for normal class in test phase,calulated as per 10% of total no
rmal class images 300.
    n_rest=10        # specifying the number of images for disease classes in test phase,calulated as per 10% of total
normal class images 100.

    # Adding the images in normal validation set by using k*n_normal to (k+1)*n_normal as index values for normal data
set divided in cell 6.
    test_normal= normal[k*n_normal:(k+1)*n_normal]
    print('------------------------Start of',k+1,'Fold-------------------------')
    print('test images for normal class from',k*n_normal,(k+1)*n_normal)

    # Adding the images in cataract validation set by using k*n_rest to (k+1)*n_rest as index values for cataract data
set divided in cell 7.
    test_cataract= cataract[k*n_rest:(k+1)*n_rest]
    print('test images for cataract class from',k*n_rest,(k+1)*n_rest)

    # Adding the images in gluacoma validation set by using k*n_rest to (k+1)*n_rest as index values for gluacoma data
set divided in cell 8.
    test_glaucoma= glaucoma[k*n_rest:(k+1)*n_rest]
    print('test images for glaucoma class from',k*n_rest,(k+1)*n_rest)

    # Adding the images in retina disease validation set by using k*n_rest to (k+1)*n_rest as index values for retina
 disease dataset divided in cell 9.
    test_retina= retina_disease[k*n_rest:(k+1)*n_rest]
    print('test images for retina disease class from',k*n_rest,(k+1)*n_rest)

    # Now for train and validation set of Normal images first adding 0 to k*n_normal images and then adding all the im
ages from (k+1)*n_normal till last image.

    train_validation_normal= normal[:k*n_normal]
    train_validation_normal= np.append(train_validation_normal,normal[(k+1)*n_normal:],axis=0)
    print('train_validation images for normal class from 0 to',k*n_normal,'and',(k+1)*n_normal,'to 300')

    # Now for train and validation set of cataract images first adding 0 to k*n_rest images and then adding all the im
ages from (k+1)*n_rest till last image.

    train_validation_cataract= cataract[:k*n_rest]
    train_validation_cataract= np.append(train_validation_cataract,cataract[(k+1)*n_rest:],axis=0)
    print('train_validation images for cataract class from 0 to',k*n_rest,'and',(k+1)*n_rest,'to 100')

    # Now for train and validation set of glaucoma images first adding 0 to k*n_rest images and then adding all the im
ages from (k+1)*n_rest till last image.
    train_validation_glaucoma= glaucoma[:k*n_rest]
    train_validation_glaucoma= np.append(train_validation_glaucoma,glaucoma[(k+1)*n_rest:],axis=0)
    print('train_validation  images for glaucoma class from 0',k*n_rest,'and',(k+1)*n_rest,'to 100')

    # Now for train and validation set of retina disease images first adding 0 to k*n_rest images and then adding all
 the images from (k+1)*n_rest till last image.
    train_validation_retina= retina_disease[:k*n_rest]
    train_validation_retina= np.append(train_validation_retina,retina_disease[(k+1)*n_rest:],axis=0)
    print('train_validation images for retina disease class from 0 to',k*n_rest,'and',(k+1)*n_rest,'to 100')


    # Splitting the train validation datasets in 80:20 ratio which would eventually give us 70% images in train and 2
0% images in validation and 10% in test.
    normal_train, normal_validation                = train_test_split(train_validation_normal, test_size=0.20, random
_state=14,shuffle=True)
    cataract_train, cataract_validation            = train_test_split(train_validation_cataract, test_size=0.20, rand
om_state=14,shuffle=True)
    glaucoma_train, glaucoma_validation            = train_test_split(train_validation_glaucoma, test_size=0.20, rand
om_state=14,shuffle=True)
    retina_disease_train, retina_disease_validation = train_test_split(train_validation_retina, test_size=0.20, random
_state=14,shuffle=True)

    # Appending all train set images for all classes
    train= np.append(normal_train,cataract_train,axis=0)
    train= np.append(train,glaucoma_train,axis=0)
    train= np.append(train,retina_disease_train,axis=0)

    # Appending all validation set images for all classes
    validation= np.append(normal_validation,cataract_validation,axis=0)
    validation= np.append(validation,glaucoma_validation,axis=0)
    validation= np.append(validation,retina_disease_validation,axis=0)

    # Appending all test set images for all classes
    test= np.append(test_normal,test_cataract,axis=0)
    test= np.append(test,test_glaucoma,axis=0)
    test= np.append(test,test_retina,axis=0)

    # Shuffling the train validation and test set as they are added sequentially.
    random.seed(6)
    np.random.shuffle(train)
    np.random.shuffle(validation)
    np.random.shuffle(test)

    # Passing the train validation test as argument for image_label_split function that return features and labels sep
arated.
```

```
    x_train,y_train,x_val,y_val,x_test,y_test = image_label_split(train,validation,test)

    # Passing the x_Train x_val and x_test as a argument for normalize function that returns the normalized and reshap
ed sets.
    x_train,x_val,x_test = normalize(x_train,x_val,x_test)

    # model building and model compile is done using a model_build_compile().

    model = model_build_compile(k)
    # passing x_train,y_train and x_val,y_val for model.fit
    history = model.fit(x_train,y_train,epochs =50, validation_data = (x_val,y_val))


    train_acc = np.append(train_acc,history.history['accuracy'])
    val_acc = np.append(val_acc,history.history['val_accuracy'])

    train_loss = np.append(train_loss,history.history['loss'])
    val_loss = np.append(val_loss,history.history['val_loss'])

    x,y,z,c = test_pred(x_test,y_test,k)

    CM.append([c])
    test_accuracy.append(x)
    test_specificity.append(y)
    test_sensitivity.append(z)
```

```
------------------------Start of 1 Fold------------------------
test images for normal class from 0 30
test images for cataract class from 0 10
test images for glaucoma class from 0 10
test images for retina disease class from 0 10
train_validation images for normal class from 0 to 0 and 30 to 300
train_validation images for cataract class from 0 to 0 and 10 to 100
train_validation  images for glaucoma class from 0 0 and 10 to 100
train_validation images for retina disease class from 0 to 0 and 10 to 100
model building and compiling for fold 1
Epoch 1/50
14/14 [==============================] - 30s 2s/step - loss: 14.8860 - accuracy: 0.4583 - val_loss: 89.6661 - val_acc
uracy: 0.2963
Epoch 2/50
14/14 [==============================] - 27s 2s/step - loss: 7.0474 - accuracy: 0.5417 - val_loss: 22.0998 - val_accu
racy: 0.3056
Epoch 3/50
14/14 [==============================] - 28s 2s/step - loss: 7.1395 - accuracy: 0.5579 - val_loss: 10.1759 - val_accu
racy: 0.3796
Epoch 4/50
14/14 [==============================] - 29s 2s/step - loss: 3.2662 - accuracy: 0.6412 - val_loss: 44.8886 - val_accu
racy: 0.2500
Epoch 5/50
14/14 [==============================] - 30s 2s/step - loss: 4.9119 - accuracy: 0.6389 - val_loss: 21.2243 - val_accu
racy: 0.1574
Epoch 6/50
14/14 [==============================] - 30s 2s/step - loss: 2.7467 - accuracy: 0.6991 - val_loss: 7.6613 - val_accur
acy: 0.3333
Epoch 7/50
14/14 [==============================] - 31s 2s/step - loss: 2.6533 - accuracy: 0.7153 - val_loss: 5.6422 - val_accur
acy: 0.5926
Epoch 8/50
14/14 [==============================] - 31s 2s/step - loss: 2.1430 - accuracy: 0.7685 - val_loss: 3.0347 - val_accur
acy: 0.5370
Epoch 9/50
14/14 [==============================] - 32s 2s/step - loss: 1.3896 - accuracy: 0.8403 - val_loss: 8.4274 - val_accur
acy: 0.2870
Epoch 10/50
14/14 [==============================] - 33s 2s/step - loss: 1.2990 - accuracy: 0.8796 - val_loss: 3.4608 - val_accur
acy: 0.6111
Epoch 11/50
14/14 [==============================] - 34s 2s/step - loss: 0.9353 - accuracy: 0.8704 - val_loss: 8.0237 - val_accur
acy: 0.3426
Epoch 12/50
14/14 [==============================] - 34s 2s/step - loss: 1.3175 - accuracy: 0.8171 - val_loss: 6.4624 - val_accur
acy: 0.3611
Epoch 13/50
14/14 [==============================] - 34s 2s/step - loss: 0.4855 - accuracy: 0.9074 - val_loss: 4.5604 - val_accur
acy: 0.4444
Epoch 14/50
14/14 [==============================] - 34s 2s/step - loss: 0.4235 - accuracy: 0.9352 - val_loss: 4.4341 - val_accur
acy: 0.4630
Epoch 15/50
14/14 [==============================] - 34s 2s/step - loss: 0.4863 - accuracy: 0.9144 - val_loss: 9.2659 - val_accur
acy: 0.4630
Epoch 16/50
14/14 [==============================] - 34s 2s/step - loss: 0.4380 - accuracy: 0.9167 - val_loss: 11.2094 - val_accu
racy: 0.3519
Epoch 17/50
14/14 [==============================] - 34s 2s/step - loss: 0.4662 - accuracy: 0.9259 - val_loss: 8.1622 - val_accur
acy: 0.5278
Epoch 18/50
14/14 [==============================] - 35s 3s/step - loss: 0.4909 - accuracy: 0.9352 - val_loss: 12.1392 - val_accu
racy: 0.4444
Epoch 19/50
14/14 [==============================] - 34s 2s/step - loss: 0.6412 - accuracy: 0.9444 - val_loss: 8.5364 - val_accur
acy: 0.4537
Epoch 20/50
14/14 [==============================] - 34s 2s/step - loss: 0.1748 - accuracy: 0.9676 - val_loss: 13.0090 - val_accu
racy: 0.2315
Epoch 21/50
14/14 [==============================] - 34s 2s/step - loss: 0.2570 - accuracy: 0.9676 - val_loss: 5.8007 - val_accur
acy: 0.5741
Epoch 22/50
14/14 [==============================] - 34s 2s/step - loss: 0.1692 - accuracy: 0.9699 - val_loss: 5.3161 - val_accur
acy: 0.5278
Epoch 23/50
14/14 [==============================] - 34s 2s/step - loss: 0.5200 - accuracy: 0.9375 - val_loss: 11.2927 - val_accu
racy: 0.4074
Epoch 24/50
14/14 [==============================] - 36s 3s/step - loss: 0.5183 - accuracy: 0.9352 - val_loss: 9.1288 - val_accur
acy: 0.5463
Epoch 25/50
14/14 [==============================] - 34s 2s/step - loss: 0.3805 - accuracy: 0.9444 - val_loss: 8.1673 - val_accur
acy: 0.6481
Epoch 26/50
14/14 [==============================] - 35s 3s/step - loss: 0.1395 - accuracy: 0.9769 - val_loss: 6.7666 - val_accur
acy: 0.6204
```

```
Epoch 27/50
14/14 [==============================] - 34s 2s/step - loss: 0.1934 - accuracy: 0.9699 - val_loss: 7.6254 - val_accur
acy: 0.6574
Epoch 28/50
14/14 [==============================] - 34s 2s/step - loss: 0.3014 - accuracy: 0.9606 - val_loss: 9.0647 - val_accur
acy: 0.5741
Epoch 29/50
14/14 [==============================] - 34s 2s/step - loss: 0.2960 - accuracy: 0.9444 - val_loss: 9.1225 - val_accur
acy: 0.6204
Epoch 30/50
14/14 [==============================] - 34s 2s/step - loss: 0.4003 - accuracy: 0.9676 - val_loss: 7.6237 - val_accur
acy: 0.5648
Epoch 31/50
14/14 [==============================] - 34s 2s/step - loss: 0.2642 - accuracy: 0.9722 - val_loss: 6.2171 - val_accur
acy: 0.6204
Epoch 32/50
14/14 [==============================] - 35s 3s/step - loss: 0.1826 - accuracy: 0.9769 - val_loss: 8.9923 - val_accur
acy: 0.5278
Epoch 33/50
14/14 [==============================] - 35s 3s/step - loss: 0.2911 - accuracy: 0.9491 - val_loss: 5.9589 - val_accur
acy: 0.6204
Epoch 34/50
14/14 [==============================] - 36s 3s/step - loss: 0.3076 - accuracy: 0.9769 - val_loss: 10.8254 - val_accu
racy: 0.5648
Epoch 35/50
14/14 [==============================] - 35s 2s/step - loss: 0.2866 - accuracy: 0.9606 - val_loss: 22.3516 - val_accu
racy: 0.5370
Epoch 36/50
14/14 [==============================] - 34s 2s/step - loss: 0.6009 - accuracy: 0.9468 - val_loss: 13.1345 - val_accu
racy: 0.5648
Epoch 37/50
14/14 [==============================] - 34s 2s/step - loss: 0.0721 - accuracy: 0.9792 - val_loss: 13.1569 - val_accu
racy: 0.5463
Epoch 38/50
14/14 [==============================] - 34s 2s/step - loss: 0.1693 - accuracy: 0.9722 - val_loss: 9.3705 - val_accur
acy: 0.5926
Epoch 39/50
14/14 [==============================] - 34s 2s/step - loss: 0.1340 - accuracy: 0.9769 - val_loss: 9.5847 - val_accur
acy: 0.4815
Epoch 40/50
14/14 [==============================] - 35s 2s/step - loss: 0.1372 - accuracy: 0.9769 - val_loss: 7.7276 - val_accur
acy: 0.5926
Epoch 41/50
14/14 [==============================] - 34s 2s/step - loss: 0.1263 - accuracy: 0.9861 - val_loss: 10.0657 - val_accu
racy: 0.4907
Epoch 42/50
14/14 [==============================] - 33s 2s/step - loss: 0.5292 - accuracy: 0.9583 - val_loss: 11.4037 - val_accu
racy: 0.6204
Epoch 43/50
14/14 [==============================] - 33s 2s/step - loss: 0.2735 - accuracy: 0.9583 - val_loss: 7.4193 - val_accur
acy: 0.5648
Epoch 44/50
14/14 [==============================] - 33s 2s/step - loss: 0.1339 - accuracy: 0.9884 - val_loss: 9.2637 - val_accur
acy: 0.6296
Epoch 45/50
14/14 [==============================] - 33s 2s/step - loss: 0.2849 - accuracy: 0.9606 - val_loss: 12.0753 - val_accu
racy: 0.5278
Epoch 46/50
14/14 [==============================] - 33s 2s/step - loss: 0.3384 - accuracy: 0.9676 - val_loss: 7.6173 - val_accur
acy: 0.5926
Epoch 47/50
14/14 [==============================] - 33s 2s/step - loss: 0.1802 - accuracy: 0.9838 - val_loss: 9.7598 - val_accur
acy: 0.5926
Epoch 48/50
14/14 [==============================] - 33s 2s/step - loss: 0.2081 - accuracy: 0.9769 - val_loss: 11.2043 - val_accu
racy: 0.5926
Epoch 49/50
14/14 [==============================] - 32s 2s/step - loss: 0.0784 - accuracy: 0.9931 - val_loss: 10.0273 - val_accu
racy: 0.5648
Epoch 50/50
14/14 [==============================] - 29s 2s/step - loss: 0.1416 - accuracy: 0.9815 - val_loss: 9.9328 - val_accur
acy: 0.6111
----------------Test accuracy for 1 fold----------------
Confusion Matrix :
 [[23  0  5  2]
 [ 3  6  1  0]
 [ 5  2  2  1]
 [ 6  4  0  0]]
Accuracy    :  0.5166666666666667
Specificity :  0.7276060277483427
Sensitivity :  0.39166666666666666
-----------------------------End of 1 Fold-------------------------
------------------------Start of 2 Fold-----------------------
test images for normal class from 30 60
test images for cataract class from 10 20
test images for glaucoma class from 10 20
test images for retina disease class from 10 20
train_validation images for normal class from 0 to 30 and 60 to 300
```

```
train_validation images for cataract class from 0 to 10 and 20 to 100
train_validation  images for glaucoma class from 0 10 and 20 to 100
train_validation images for retina disease class from 0 to 10 and 20 to 100
model building and compiling for fold 2
Epoch 1/50
14/14 [==============================] - 34s 2s/step - loss: 15.4808 - accuracy: 0.4074 - val_loss: 101.0422 - val_ac
curacy: 0.1944
Epoch 2/50
14/14 [==============================] - 29s 2s/step - loss: 7.2575 - accuracy: 0.5694 - val_loss: 25.8822 - val_accu
racy: 0.5000
Epoch 3/50
14/14 [==============================] - 30s 2s/step - loss: 4.9677 - accuracy: 0.6319 - val_loss: 36.0923 - val_accu
racy: 0.4352
Epoch 4/50
14/14 [==============================] - 30s 2s/step - loss: 5.3211 - accuracy: 0.6227 - val_loss: 16.2035 - val_accu
racy: 0.3148
Epoch 5/50
14/14 [==============================] - 31s 2s/step - loss: 2.3461 - accuracy: 0.7176 - val_loss: 11.3036 - val_accu
racy: 0.3704
Epoch 6/50
14/14 [==============================] - 32s 2s/step - loss: 2.1307 - accuracy: 0.7546 - val_loss: 10.7333 - val_accu
racy: 0.4167
Epoch 7/50
14/14 [==============================] - 33s 2s/step - loss: 3.3924 - accuracy: 0.7477 - val_loss: 12.2236 - val_accu
racy: 0.3426
Epoch 8/50
14/14 [==============================] - 32s 2s/step - loss: 1.3359 - accuracy: 0.8125 - val_loss: 13.2954 - val_accu
racy: 0.2963
Epoch 9/50
14/14 [==============================] - 32s 2s/step - loss: 0.7743 - accuracy: 0.8356 - val_loss: 6.7182 - val_accur
acy: 0.3981
Epoch 10/50
14/14 [==============================] - 32s 2s/step - loss: 0.7253 - accuracy: 0.8773 - val_loss: 10.9105 - val_accu
racy: 0.2963
Epoch 11/50
14/14 [==============================] - 33s 2s/step - loss: 1.6554 - accuracy: 0.7986 - val_loss: 20.3839 - val_accu
racy: 0.2593
Epoch 12/50
14/14 [==============================] - 32s 2s/step - loss: 0.6846 - accuracy: 0.8843 - val_loss: 6.1565 - val_accur
acy: 0.5000
Epoch 13/50
14/14 [==============================] - 33s 2s/step - loss: 0.4439 - accuracy: 0.9306 - val_loss: 6.2114 - val_accur
acy: 0.4907
Epoch 14/50
14/14 [==============================] - 32s 2s/step - loss: 0.7279 - accuracy: 0.9005 - val_loss: 11.9120 - val_accu
racy: 0.5185
Epoch 15/50
14/14 [==============================] - 32s 2s/step - loss: 0.6463 - accuracy: 0.9120 - val_loss: 6.0836 - val_accur
acy: 0.5463
Epoch 16/50
14/14 [==============================] - 32s 2s/step - loss: 0.6228 - accuracy: 0.9144 - val_loss: 9.4528 - val_accur
acy: 0.4815
Epoch 17/50
14/14 [==============================] - 33s 2s/step - loss: 0.6955 - accuracy: 0.9190 - val_loss: 7.9997 - val_accur
acy: 0.4907
Epoch 18/50
14/14 [==============================] - 32s 2s/step - loss: 0.2401 - accuracy: 0.9606 - val_loss: 7.0817 - val_accur
acy: 0.5185
Epoch 19/50
14/14 [==============================] - 33s 2s/step - loss: 0.1899 - accuracy: 0.9676 - val_loss: 15.1554 - val_accu
racy: 0.3148
Epoch 20/50
14/14 [==============================] - 33s 2s/step - loss: 0.5640 - accuracy: 0.9144 - val_loss: 9.6509 - val_accur
acy: 0.6204
Epoch 21/50
14/14 [==============================] - 33s 2s/step - loss: 0.3309 - accuracy: 0.9491 - val_loss: 7.4340 - val_accur
acy: 0.4352
Epoch 22/50
14/14 [==============================] - 32s 2s/step - loss: 0.2860 - accuracy: 0.9444 - val_loss: 9.9594 - val_accur
acy: 0.5741
Epoch 23/50
14/14 [==============================] - 32s 2s/step - loss: 0.5936 - accuracy: 0.9468 - val_loss: 6.7852 - val_accur
acy: 0.5370
Epoch 24/50
14/14 [==============================] - 33s 2s/step - loss: 0.2267 - accuracy: 0.9606 - val_loss: 8.6874 - val_accur
acy: 0.5093
Epoch 25/50
14/14 [==============================] - 32s 2s/step - loss: 0.2460 - accuracy: 0.9630 - val_loss: 7.8742 - val_accur
acy: 0.5741
Epoch 26/50
14/14 [==============================] - 32s 2s/step - loss: 0.1751 - accuracy: 0.9792 - val_loss: 8.0666 - val_accur
acy: 0.5370
Epoch 27/50
14/14 [==============================] - 34s 2s/step - loss: 0.7177 - accuracy: 0.9491 - val_loss: 9.5184 - val_accur
acy: 0.6019
Epoch 28/50
14/14 [==============================] - 33s 2s/step - loss: 0.3423 - accuracy: 0.9560 - val_loss: 15.9977 - val_accu
racy: 0.5926
```

```
Epoch 29/50
14/14 [==============================] - 34s 2s/step - loss: 0.1891 - accuracy: 0.9676 - val_loss: 12.8151 - val_accu
racy: 0.6019
Epoch 30/50
14/14 [==============================] - 33s 2s/step - loss: 0.3371 - accuracy: 0.9491 - val_loss: 12.2592 - val_accu
racy: 0.5926
Epoch 31/50
14/14 [==============================] - 32s 2s/step - loss: 0.4045 - accuracy: 0.9375 - val_loss: 9.9118 - val_accur
acy: 0.5185
Epoch 32/50
14/14 [==============================] - 33s 2s/step - loss: 0.0997 - accuracy: 0.9815 - val_loss: 12.1499 - val_accu
racy: 0.5648
Epoch 33/50
14/14 [==============================] - 33s 2s/step - loss: 0.1822 - accuracy: 0.9653 - val_loss: 13.3021 - val_accu
racy: 0.4444
Epoch 34/50
14/14 [==============================] - 33s 2s/step - loss: 0.0895 - accuracy: 0.9838 - val_loss: 17.9640 - val_accu
racy: 0.4167
Epoch 35/50
14/14 [==============================] - 33s 2s/step - loss: 0.6366 - accuracy: 0.9398 - val_loss: 21.3324 - val_accu
racy: 0.3241
Epoch 36/50
14/14 [==============================] - 33s 2s/step - loss: 0.1396 - accuracy: 0.9745 - val_loss: 23.8017 - val_accu
racy: 0.5370
Epoch 37/50
14/14 [==============================] - 33s 2s/step - loss: 0.2164 - accuracy: 0.9699 - val_loss: 22.1754 - val_accu
racy: 0.4352
Epoch 38/50
14/14 [==============================] - 33s 2s/step - loss: 0.3302 - accuracy: 0.9676 - val_loss: 29.8690 - val_accu
racy: 0.5648
Epoch 39/50
14/14 [==============================] - 33s 2s/step - loss: 0.0820 - accuracy: 0.9861 - val_loss: 23.5094 - val_accu
racy: 0.5741
Epoch 40/50
14/14 [==============================] - 33s 2s/step - loss: 0.1494 - accuracy: 0.9745 - val_loss: 56.6338 - val_accu
racy: 0.5370
Epoch 41/50
14/14 [==============================] - 33s 2s/step - loss: 0.1768 - accuracy: 0.9838 - val_loss: 49.6517 - val_accu
racy: 0.4907
Epoch 42/50
14/14 [==============================] - 33s 2s/step - loss: 0.1908 - accuracy: 0.9838 - val_loss: 44.1146 - val_accu
racy: 0.5463
Epoch 43/50
14/14 [==============================] - 33s 2s/step - loss: 0.1900 - accuracy: 0.9630 - val_loss: 46.5102 - val_accu
racy: 0.5741
Epoch 44/50
14/14 [==============================] - 33s 2s/step - loss: 0.0377 - accuracy: 0.9838 - val_loss: 71.9868 - val_accu
racy: 0.5556
Epoch 45/50
14/14 [==============================] - 33s 2s/step - loss: 0.2200 - accuracy: 0.9630 - val_loss: 57.3072 - val_accu
racy: 0.6019
Epoch 46/50
14/14 [==============================] - 33s 2s/step - loss: 0.0598 - accuracy: 0.9815 - val_loss: 45.0992 - val_accu
racy: 0.5833
Epoch 47/50
14/14 [==============================] - 33s 2s/step - loss: 0.0908 - accuracy: 0.9838 - val_loss: 28.8695 - val_accu
racy: 0.5741
Epoch 48/50
14/14 [==============================] - 34s 2s/step - loss: 0.1189 - accuracy: 0.9861 - val_loss: 10.3059 - val_accu
racy: 0.5926
Epoch 49/50
14/14 [==============================] - 33s 2s/step - loss: 0.1719 - accuracy: 0.9792 - val_loss: 36.2556 - val_accu
racy: 0.5556
Epoch 50/50
14/14 [==============================] - 33s 2s/step - loss: 0.0268 - accuracy: 0.9931 - val_loss: 28.1140 - val_accu
racy: 0.5926
----------------Test accuracy for 2 fold----------------
Confusion Matrix :
 [[26  0  3  1]
 [ 4  5  1  0]
 [ 6  0  3  1]
 [ 6  0  1  3]]
Accuracy    :  0.6166666666666667
Specificity :  0.8059116809116809
Sensitivity :  0.4916666666666667
----------------------------End of 2 Fold-------------------------
------------------------Start of 3 Fold-----------------------
test images for normal class from 60 90
test images for cataract class from 20 30
test images for glaucoma class from 20 30
test images for retina disease class from 20 30
train_validation images for normal class from 0 to 60 and 90 to 300
train_validation images for cataract class from 0 to 20 and 30 to 100
train_validation  images for glaucoma class from 0 20 and 30 to 100
train_validation images for retina disease class from 0 to 20 and 30 to 100
model building and compiling for fold 3
Epoch 1/50
14/14 [==============================] - 36s 2s/step - loss: 18.2885 - accuracy: 0.4120 - val_loss: 55.8603 - val_acc
```

```
uracy: 0.2500
Epoch 2/50
14/14 [==============================] - 30s 2s/step - loss: 8.4935 - accuracy: 0.4699 - val_loss: 25.6996 - val_accu
racy: 0.4815
Epoch 3/50
14/14 [==============================] - 31s 2s/step - loss: 4.8749 - accuracy: 0.6250 - val_loss: 36.7543 - val_accu
racy: 0.1852
Epoch 4/50
14/14 [==============================] - 32s 2s/step - loss: 6.2817 - accuracy: 0.5324 - val_loss: 18.6895 - val_accu
racy: 0.1852
Epoch 5/50
14/14 [==============================] - 32s 2s/step - loss: 3.2682 - accuracy: 0.6782 - val_loss: 6.2715 - val_accur
acy: 0.4259
Epoch 6/50
14/14 [==============================] - 34s 2s/step - loss: 4.0977 - accuracy: 0.6644 - val_loss: 7.8782 - val_accur
acy: 0.3704
Epoch 7/50
14/14 [==============================] - 33s 2s/step - loss: 2.0204 - accuracy: 0.7407 - val_loss: 12.0657 - val_accu
racy: 0.2593
Epoch 8/50
14/14 [==============================] - 31s 2s/step - loss: 1.7862 - accuracy: 0.7731 - val_loss: 10.9524 - val_accu
racy: 0.2963
Epoch 9/50
14/14 [==============================] - 29s 2s/step - loss: 1.3591 - accuracy: 0.8171 - val_loss: 7.0844 - val_accur
acy: 0.4444
Epoch 10/50
14/14 [==============================] - 29s 2s/step - loss: 1.1874 - accuracy: 0.8287 - val_loss: 5.9176 - val_accur
acy: 0.3704
Epoch 11/50
14/14 [==============================] - 30s 2s/step - loss: 1.4585 - accuracy: 0.8588 - val_loss: 9.9811 - val_accur
acy: 0.4630
Epoch 12/50
14/14 [==============================] - 29s 2s/step - loss: 0.6893 - accuracy: 0.8843 - val_loss: 12.8085 - val_accu
racy: 0.2130
Epoch 13/50
14/14 [==============================] - 26s 2s/step - loss: 0.8930 - accuracy: 0.8750 - val_loss: 5.9862 - val_accur
acy: 0.4352
Epoch 14/50
14/14 [==============================] - 26s 2s/step - loss: 0.5918 - accuracy: 0.8958 - val_loss: 5.7060 - val_accur
acy: 0.4352
Epoch 15/50
14/14 [==============================] - 27s 2s/step - loss: 0.5035 - accuracy: 0.9074 - val_loss: 6.7806 - val_accur
acy: 0.5370
Epoch 16/50
14/14 [==============================] - 30s 2s/step - loss: 0.7311 - accuracy: 0.8796 - val_loss: 8.7960 - val_accur
acy: 0.4074
Epoch 17/50
14/14 [==============================] - 31s 2s/step - loss: 0.4908 - accuracy: 0.9282 - val_loss: 9.1816 - val_accur
acy: 0.2685
Epoch 18/50
14/14 [==============================] - 31s 2s/step - loss: 0.5775 - accuracy: 0.9144 - val_loss: 14.6239 - val_accu
racy: 0.3333
Epoch 19/50
14/14 [==============================] - 32s 2s/step - loss: 0.7790 - accuracy: 0.9630 - val_loss: 8.2559 - val_accur
acy: 0.3889
Epoch 20/50
14/14 [==============================] - 33s 2s/step - loss: 0.4217 - accuracy: 0.9236 - val_loss: 6.4400 - val_accur
acy: 0.4444
Epoch 21/50
14/14 [==============================] - 34s 2s/step - loss: 0.5040 - accuracy: 0.9375 - val_loss: 7.7507 - val_accur
acy: 0.5833
Epoch 22/50
14/14 [==============================] - 34s 2s/step - loss: 0.2900 - accuracy: 0.9560 - val_loss: 5.2420 - val_accur
acy: 0.5185
Epoch 23/50
14/14 [==============================] - 34s 2s/step - loss: 0.2625 - accuracy: 0.9537 - val_loss: 5.6834 - val_accur
acy: 0.6111
Epoch 24/50
14/14 [==============================] - 34s 2s/step - loss: 0.2769 - accuracy: 0.9653 - val_loss: 5.6531 - val_accur
acy: 0.5278
Epoch 25/50
14/14 [==============================] - 34s 2s/step - loss: 0.2640 - accuracy: 0.9560 - val_loss: 8.6570 - val_accur
acy: 0.4444
Epoch 26/50
14/14 [==============================] - 34s 2s/step - loss: 1.2681 - accuracy: 0.9144 - val_loss: 6.9777 - val_accur
acy: 0.5741
Epoch 27/50
14/14 [==============================] - 34s 2s/step - loss: 0.0523 - accuracy: 0.9884 - val_loss: 6.2912 - val_accur
acy: 0.6019
Epoch 28/50
14/14 [==============================] - 34s 2s/step - loss: 0.3618 - accuracy: 0.9699 - val_loss: 9.2709 - val_accur
acy: 0.5185
Epoch 29/50
14/14 [==============================] - 34s 2s/step - loss: 0.1843 - accuracy: 0.9676 - val_loss: 9.1950 - val_accur
acy: 0.4630
Epoch 30/50
14/14 [==============================] - 34s 2s/step - loss: 0.3368 - accuracy: 0.9583 - val_loss: 11.8202 - val_accu
racy: 0.4259
```

```
Epoch 31/50
14/14 [==============================] - 34s 2s/step - loss: 0.5444 - accuracy: 0.9514 - val_loss: 9.3025 - val_accur
acy: 0.5000
Epoch 32/50
14/14 [==============================] - 35s 3s/step - loss: 0.1854 - accuracy: 0.9745 - val_loss: 10.2256 - val_accu
racy: 0.4352
Epoch 33/50
14/14 [==============================] - 34s 2s/step - loss: 0.1961 - accuracy: 0.9630 - val_loss: 11.3140 - val_accu
racy: 0.4167
Epoch 34/50
14/14 [==============================] - 34s 2s/step - loss: 0.2022 - accuracy: 0.9792 - val_loss: 9.6994 - val_accur
acy: 0.4444
Epoch 35/50
14/14 [==============================] - 34s 2s/step - loss: 0.1810 - accuracy: 0.9699 - val_loss: 10.0850 - val_accu
racy: 0.4630
Epoch 36/50
14/14 [==============================] - 34s 2s/step - loss: 0.1280 - accuracy: 0.9792 - val_loss: 13.0794 - val_accu
racy: 0.3796
Epoch 37/50
14/14 [==============================] - 34s 2s/step - loss: 0.3372 - accuracy: 0.9676 - val_loss: 8.4258 - val_accur
acy: 0.4907
Epoch 38/50
14/14 [==============================] - 34s 2s/step - loss: 0.0362 - accuracy: 0.9861 - val_loss: 8.1735 - val_accur
acy: 0.5556
Epoch 39/50
14/14 [==============================] - 34s 2s/step - loss: 0.1203 - accuracy: 0.9884 - val_loss: 8.9805 - val_accur
acy: 0.4907
Epoch 40/50
14/14 [==============================] - 34s 2s/step - loss: 0.2786 - accuracy: 0.9630 - val_loss: 9.7688 - val_accur
acy: 0.5556
Epoch 41/50
14/14 [==============================] - 35s 3s/step - loss: 0.1894 - accuracy: 0.9769 - val_loss: 9.0422 - val_accur
acy: 0.6019
Epoch 42/50
14/14 [==============================] - 34s 2s/step - loss: 0.2131 - accuracy: 0.9722 - val_loss: 17.8178 - val_accu
racy: 0.5741
Epoch 43/50
14/14 [==============================] - 35s 3s/step - loss: 0.2740 - accuracy: 0.9722 - val_loss: 9.4403 - val_accur
acy: 0.5833
Epoch 44/50
14/14 [==============================] - 34s 2s/step - loss: 0.0550 - accuracy: 0.9884 - val_loss: 11.3925 - val_accu
racy: 0.6019
Epoch 45/50
14/14 [==============================] - 34s 2s/step - loss: 0.2205 - accuracy: 0.9583 - val_loss: 13.0648 - val_accu
racy: 0.6111
Epoch 46/50
14/14 [==============================] - 34s 2s/step - loss: 0.0594 - accuracy: 0.9861 - val_loss: 13.1816 - val_accu
racy: 0.5556
Epoch 47/50
14/14 [==============================] - 34s 2s/step - loss: 0.0626 - accuracy: 0.9815 - val_loss: 15.8847 - val_accu
racy: 0.3796
Epoch 48/50
14/14 [==============================] - 34s 2s/step - loss: 0.3336 - accuracy: 0.9583 - val_loss: 11.0012 - val_accu
racy: 0.5741
Epoch 49/50
14/14 [==============================] - 35s 2s/step - loss: 0.0676 - accuracy: 0.9884 - val_loss: 10.2444 - val_accu
racy: 0.5741
Epoch 50/50
14/14 [==============================] - 34s 2s/step - loss: 0.1219 - accuracy: 0.9931 - val_loss: 12.8790 - val_accu
racy: 0.6111
----------------Test accuracy for 3 fold----------------
Confusion Matrix :
 [[24  0  5  1]
 [ 2  7  0  1]
 [ 4  0  5  1]
 [ 6  0  1  3]]
Accuracy    :  0.65
Specificity :  0.8321581196581196
Sensitivity :  0.575
-----------------------------End of 3 Fold--------------------------
------------------------Start of 4 Fold-----------------------
test images for normal class from 90 120
test images for cataract class from 30 40
test images for glaucoma class from 30 40
test images for retina disease class from 30 40
train_validation images for normal class from 0 to 90 and 120 to 300
train_validation images for cataract class from 0 to 30 and 40 to 100
train_validation  images for glaucoma class from 0 30 and 40 to 100
train_validation images for retina disease class from 0 to 30 and 40 to 100
model building and compiling for fold 4
Epoch 1/50
14/14 [==============================] - 40s 3s/step - loss: 17.6823 - accuracy: 0.4190 - val_loss: 88.9815 - val_acc
uracy: 0.3611
Epoch 2/50
14/14 [==============================] - 34s 2s/step - loss: 8.0115 - accuracy: 0.5116 - val_loss: 54.8419 - val_accu
racy: 0.2037
Epoch 3/50
14/14 [==============================] - 34s 2s/step - loss: 5.9848 - accuracy: 0.5718 - val_loss: 35.0940 - val_accu
```

```
racy: 0.2500
Epoch 4/50
14/14 [==============================] - 34s 2s/step - loss: 4.9056 - accuracy: 0.6227 - val_loss: 21.2603 - val_accu
racy: 0.3796
Epoch 5/50
14/14 [==============================] - 34s 2s/step - loss: 4.6078 - accuracy: 0.6481 - val_loss: 6.2023 - val_accur
acy: 0.4167
Epoch 6/50
14/14 [==============================] - 34s 2s/step - loss: 1.9845 - accuracy: 0.7269 - val_loss: 21.8392 - val_accu
racy: 0.2870
Epoch 7/50
14/14 [==============================] - 34s 2s/step - loss: 2.4932 - accuracy: 0.7292 - val_loss: 10.8939 - val_accu
racy: 0.5741
Epoch 8/50
14/14 [==============================] - 34s 2s/step - loss: 2.0287 - accuracy: 0.7477 - val_loss: 6.9370 - val_accur
acy: 0.3333
Epoch 9/50
14/14 [==============================] - 35s 3s/step - loss: 1.2005 - accuracy: 0.8287 - val_loss: 6.9803 - val_accur
acy: 0.3056
Epoch 10/50
14/14 [==============================] - 35s 2s/step - loss: 0.8895 - accuracy: 0.8426 - val_loss: 5.5220 - val_accur
acy: 0.2963
Epoch 11/50
14/14 [==============================] - 35s 2s/step - loss: 0.7213 - accuracy: 0.8866 - val_loss: 5.3877 - val_accur
acy: 0.5833
Epoch 12/50
14/14 [==============================] - 34s 2s/step - loss: 1.2538 - accuracy: 0.9074 - val_loss: 6.6584 - val_accur
acy: 0.5648
Epoch 13/50
14/14 [==============================] - 35s 2s/step - loss: 0.4347 - accuracy: 0.9329 - val_loss: 7.1323 - val_accur
acy: 0.5926
Epoch 14/50
14/14 [==============================] - 34s 2s/step - loss: 0.6569 - accuracy: 0.8935 - val_loss: 7.7552 - val_accur
acy: 0.6019
Epoch 15/50
14/14 [==============================] - 34s 2s/step - loss: 0.4263 - accuracy: 0.9236 - val_loss: 7.9868 - val_accur
acy: 0.4907
Epoch 16/50
14/14 [==============================] - 36s 3s/step - loss: 0.3754 - accuracy: 0.9398 - val_loss: 5.3309 - val_accur
acy: 0.6204
Epoch 17/50
14/14 [==============================] - 34s 2s/step - loss: 0.6407 - accuracy: 0.9236 - val_loss: 7.2293 - val_accur
acy: 0.5463
Epoch 18/50
14/14 [==============================] - 34s 2s/step - loss: 0.4179 - accuracy: 0.9468 - val_loss: 5.2832 - val_accur
acy: 0.4630
Epoch 19/50
14/14 [==============================] - 35s 3s/step - loss: 0.9704 - accuracy: 0.8843 - val_loss: 5.7517 - val_accur
acy: 0.5463
Epoch 20/50
14/14 [==============================] - 34s 2s/step - loss: 0.4711 - accuracy: 0.9329 - val_loss: 6.4703 - val_accur
acy: 0.4259
Epoch 21/50
14/14 [==============================] - 34s 2s/step - loss: 0.2885 - accuracy: 0.9560 - val_loss: 4.6769 - val_accur
acy: 0.4259
Epoch 22/50
14/14 [==============================] - 34s 2s/step - loss: 0.1252 - accuracy: 0.9560 - val_loss: 6.1856 - val_accur
acy: 0.5741
Epoch 23/50
14/14 [==============================] - 34s 2s/step - loss: 0.2176 - accuracy: 0.9630 - val_loss: 9.5491 - val_accur
acy: 0.4815
Epoch 24/50
14/14 [==============================] - 35s 2s/step - loss: 0.3912 - accuracy: 0.9421 - val_loss: 7.5385 - val_accur
acy: 0.4907
Epoch 25/50
14/14 [==============================] - 34s 2s/step - loss: 0.2390 - accuracy: 0.9815 - val_loss: 8.4291 - val_accur
acy: 0.5278
Epoch 26/50
14/14 [==============================] - 34s 2s/step - loss: 0.3330 - accuracy: 0.9421 - val_loss: 6.2748 - val_accur
acy: 0.6296
Epoch 27/50
14/14 [==============================] - 35s 2s/step - loss: 0.2255 - accuracy: 0.9676 - val_loss: 7.3197 - val_accur
acy: 0.4167
Epoch 28/50
14/14 [==============================] - 35s 2s/step - loss: 0.1715 - accuracy: 0.9699 - val_loss: 11.3137 - val_accu
racy: 0.3611
Epoch 29/50
14/14 [==============================] - 34s 2s/step - loss: 0.3337 - accuracy: 0.9537 - val_loss: 8.4697 - val_accur
acy: 0.5463
Epoch 30/50
14/14 [==============================] - 34s 2s/step - loss: 0.2549 - accuracy: 0.9514 - val_loss: 6.8642 - val_accur
acy: 0.6481
Epoch 31/50
14/14 [==============================] - 34s 2s/step - loss: 0.1280 - accuracy: 0.9722 - val_loss: 6.8071 - val_accur
acy: 0.6204
Epoch 32/50
14/14 [==============================] - 34s 2s/step - loss: 0.1313 - accuracy: 0.9745 - val_loss: 7.2856 - val_accur
acy: 0.6111
```

```
Epoch 33/50
14/14 [==============================] - 34s 2s/step - loss: 0.1248 - accuracy: 0.9745 - val_loss: 6.9703 - val_accur
acy: 0.6296
Epoch 34/50
14/14 [==============================] - 34s 2s/step - loss: 0.1606 - accuracy: 0.9653 - val_loss: 16.3538 - val_accu
racy: 0.6111
Epoch 35/50
14/14 [==============================] - 34s 2s/step - loss: 0.3086 - accuracy: 0.9583 - val_loss: 6.3320 - val_accur
acy: 0.5833
Epoch 36/50
14/14 [==============================] - 34s 2s/step - loss: 0.1019 - accuracy: 0.9792 - val_loss: 6.9897 - val_accur
acy: 0.5648
Epoch 37/50
14/14 [==============================] - 34s 2s/step - loss: 0.2555 - accuracy: 0.9583 - val_loss: 6.8429 - val_accur
acy: 0.5833
Epoch 38/50
14/14 [==============================] - 34s 2s/step - loss: 0.0816 - accuracy: 0.9815 - val_loss: 7.8798 - val_accur
acy: 0.5648
Epoch 39/50
14/14 [==============================] - 35s 2s/step - loss: 0.4104 - accuracy: 0.9606 - val_loss: 9.7804 - val_accur
acy: 0.5926
Epoch 40/50
14/14 [==============================] - 34s 2s/step - loss: 0.2326 - accuracy: 0.9630 - val_loss: 9.0354 - val_accur
acy: 0.6204
Epoch 41/50
14/14 [==============================] - 34s 2s/step - loss: 0.0113 - accuracy: 0.9954 - val_loss: 7.5450 - val_accur
acy: 0.6296
Epoch 42/50
14/14 [==============================] - 34s 2s/step - loss: 0.1655 - accuracy: 0.9722 - val_loss: 8.6308 - val_accur
acy: 0.4537
Epoch 43/50
14/14 [==============================] - 34s 2s/step - loss: 0.0822 - accuracy: 0.9884 - val_loss: 5.3833 - val_accur
acy: 0.5278
Epoch 44/50
14/14 [==============================] - 34s 2s/step - loss: 0.0960 - accuracy: 0.9769 - val_loss: 7.2771 - val_accur
acy: 0.5278
Epoch 45/50
14/14 [==============================] - 34s 2s/step - loss: 0.0473 - accuracy: 0.9907 - val_loss: 7.1084 - val_accur
acy: 0.5278
Epoch 46/50
14/14 [==============================] - 34s 2s/step - loss: 0.1571 - accuracy: 0.9792 - val_loss: 6.7730 - val_accur
acy: 0.5370
Epoch 47/50
14/14 [==============================] - 34s 2s/step - loss: 0.1106 - accuracy: 0.9838 - val_loss: 9.3470 - val_accur
acy: 0.5000
Epoch 48/50
14/14 [==============================] - 34s 2s/step - loss: 0.2330 - accuracy: 0.9722 - val_loss: 7.4826 - val_accur
acy: 0.5833
Epoch 49/50
14/14 [==============================] - 34s 2s/step - loss: 0.0384 - accuracy: 0.9884 - val_loss: 8.2313 - val_accur
acy: 0.5648
Epoch 50/50
14/14 [==============================] - 34s 2s/step - loss: 0.0954 - accuracy: 0.9861 - val_loss: 9.4985 - val_accur
acy: 0.4537
----------------Test accuracy for 4 fold----------------
Confusion Matrix :
 [[10  0  7 13]
 [ 1  4  3  2]
 [ 0  2  5  3]
 [ 2  3  3  2]]
Accuracy    :  0.35
Specificity :  0.6559198024715266
Sensitivity :  0.35833333333333334
----------------------------End of 4 Fold--------------------------
------------------------Start of 5 Fold-----------------------
test images for normal class from 120 150
test images for cataract class from 40 50
test images for glaucoma class from 40 50
test images for retina disease class from 40 50
train_validation images for normal class from 0 to 120 and 150 to 300
train_validation images for cataract class from 0 to 40 and 50 to 100
train_validation  images for glaucoma class from 0 40 and 50 to 100
train_validation images for retina disease class from 0 to 40 and 50 to 100
model building and compiling for fold 5
Epoch 1/50
14/14 [==============================] - 40s 3s/step - loss: 16.4355 - accuracy: 0.4259 - val_loss: 61.9567 - val_acc
uracy: 0.5278
Epoch 2/50
14/14 [==============================] - 34s 2s/step - loss: 7.5424 - accuracy: 0.5046 - val_loss: 13.8406 - val_accu
racy: 0.5278
Epoch 3/50
14/14 [==============================] - 34s 2s/step - loss: 5.1361 - accuracy: 0.5787 - val_loss: 43.1406 - val_accu
racy: 0.1574
Epoch 4/50
14/14 [==============================] - 34s 2s/step - loss: 5.8621 - accuracy: 0.5787 - val_loss: 13.8491 - val_accu
racy: 0.2963
Epoch 5/50
14/14 [==============================] - 34s 2s/step - loss: 3.1888 - accuracy: 0.6458 - val_loss: 18.5413 - val_accu
```

```
racy: 0.4352
Epoch 6/50
14/14 [==============================] - 35s 3s/step - loss: 2.1968 - accuracy: 0.7130 - val_loss: 10.0898 - val_accu
racy: 0.4815
Epoch 7/50
14/14 [==============================] - 34s 2s/step - loss: 3.7607 - accuracy: 0.7014 - val_loss: 4.9406 - val_accur
acy: 0.5278
Epoch 8/50
14/14 [==============================] - 34s 2s/step - loss: 1.8838 - accuracy: 0.7546 - val_loss: 6.4885 - val_accur
acy: 0.4630
Epoch 9/50
14/14 [==============================] - 34s 2s/step - loss: 0.7088 - accuracy: 0.8634 - val_loss: 6.2321 - val_accur
acy: 0.3889
Epoch 10/50
14/14 [==============================] - 35s 2s/step - loss: 1.9164 - accuracy: 0.8194 - val_loss: 3.8336 - val_accur
acy: 0.4907
Epoch 11/50
14/14 [==============================] - 34s 2s/step - loss: 0.3955 - accuracy: 0.9190 - val_loss: 3.6902 - val_accur
acy: 0.5556
Epoch 12/50
14/14 [==============================] - 35s 3s/step - loss: 0.9235 - accuracy: 0.8657 - val_loss: 6.6337 - val_accur
acy: 0.4722
Epoch 13/50
14/14 [==============================] - 34s 2s/step - loss: 0.4260 - accuracy: 0.9236 - val_loss: 5.9793 - val_accur
acy: 0.5000
Epoch 14/50
14/14 [==============================] - 35s 3s/step - loss: 0.5287 - accuracy: 0.9097 - val_loss: 11.9845 - val_accu
racy: 0.3519
Epoch 15/50
14/14 [==============================] - 36s 3s/step - loss: 0.5756 - accuracy: 0.9236 - val_loss: 11.2627 - val_accu
racy: 0.4537
Epoch 16/50
14/14 [==============================] - 34s 2s/step - loss: 0.9804 - accuracy: 0.9537 - val_loss: 6.1248 - val_accur
acy: 0.5741
Epoch 17/50
14/14 [==============================] - 34s 2s/step - loss: 0.6296 - accuracy: 0.9028 - val_loss: 5.5627 - val_accur
acy: 0.5926
Epoch 18/50
14/14 [==============================] - 34s 2s/step - loss: 0.1439 - accuracy: 0.9699 - val_loss: 7.5224 - val_accur
acy: 0.5185
Epoch 19/50
14/14 [==============================] - 34s 2s/step - loss: 0.6905 - accuracy: 0.9097 - val_loss: 17.2407 - val_accu
racy: 0.3056
Epoch 20/50
14/14 [==============================] - 35s 2s/step - loss: 0.6323 - accuracy: 0.9352 - val_loss: 11.4356 - val_accu
racy: 0.3056
Epoch 21/50
14/14 [==============================] - 35s 3s/step - loss: 0.4049 - accuracy: 0.9352 - val_loss: 5.6127 - val_accur
acy: 0.5556
Epoch 22/50
14/14 [==============================] - 35s 2s/step - loss: 0.1730 - accuracy: 0.9606 - val_loss: 5.2137 - val_accur
acy: 0.5648
Epoch 23/50
14/14 [==============================] - 34s 2s/step - loss: 0.3369 - accuracy: 0.9606 - val_loss: 11.6057 - val_accu
racy: 0.4352
Epoch 24/50
14/14 [==============================] - 35s 2s/step - loss: 0.4040 - accuracy: 0.9653 - val_loss: 10.2403 - val_accu
racy: 0.6019
Epoch 25/50
14/14 [==============================] - 34s 2s/step - loss: 0.4579 - accuracy: 0.9352 - val_loss: 10.9499 - val_accu
racy: 0.5556
Epoch 26/50
14/14 [==============================] - 35s 3s/step - loss: 0.6719 - accuracy: 0.9352 - val_loss: 7.9195 - val_accur
acy: 0.5833
Epoch 27/50
14/14 [==============================] - 34s 2s/step - loss: 0.2101 - accuracy: 0.9514 - val_loss: 9.6943 - val_accur
acy: 0.5926
Epoch 28/50
14/14 [==============================] - 34s 2s/step - loss: 0.2897 - accuracy: 0.9653 - val_loss: 8.2394 - val_accur
acy: 0.4630
Epoch 29/50
14/14 [==============================] - 34s 2s/step - loss: 0.0393 - accuracy: 0.9884 - val_loss: 7.3633 - val_accur
acy: 0.5648
Epoch 30/50
14/14 [==============================] - 35s 3s/step - loss: 0.0354 - accuracy: 0.9907 - val_loss: 8.1126 - val_accur
acy: 0.5185
Epoch 31/50
14/14 [==============================] - 35s 3s/step - loss: 0.3244 - accuracy: 0.9606 - val_loss: 7.5362 - val_accur
acy: 0.5556
Epoch 32/50
14/14 [==============================] - 35s 2s/step - loss: 0.3092 - accuracy: 0.9606 - val_loss: 12.0642 - val_accu
racy: 0.3889
Epoch 33/50
14/14 [==============================] - 34s 2s/step - loss: 0.2183 - accuracy: 0.9699 - val_loss: 7.6610 - val_accur
acy: 0.5556
Epoch 34/50
14/14 [==============================] - 34s 2s/step - loss: 0.2960 - accuracy: 0.9537 - val_loss: 12.7373 - val_accu
racy: 0.4259
```

```
Epoch 35/50
14/14 [==============================] - 34s 2s/step - loss: 0.0682 - accuracy: 0.9861 - val_loss: 11.5338 - val_accu
racy: 0.4537
Epoch 36/50
14/14 [==============================] - 34s 2s/step - loss: 0.0688 - accuracy: 0.9907 - val_loss: 10.3831 - val_accu
racy: 0.4537
Epoch 37/50
14/14 [==============================] - 34s 2s/step - loss: 0.3435 - accuracy: 0.9491 - val_loss: 16.0527 - val_accu
racy: 0.5278
Epoch 38/50
14/14 [==============================] - 35s 3s/step - loss: 0.1149 - accuracy: 0.9769 - val_loss: 16.9993 - val_accu
racy: 0.5741
Epoch 39/50
14/14 [==============================] - 34s 2s/step - loss: 0.1966 - accuracy: 0.9815 - val_loss: 15.3881 - val_accu
racy: 0.5278
Epoch 40/50
14/14 [==============================] - 34s 2s/step - loss: 0.1557 - accuracy: 0.9792 - val_loss: 15.9155 - val_accu
racy: 0.5648
Epoch 41/50
14/14 [==============================] - 35s 2s/step - loss: 0.2958 - accuracy: 0.9653 - val_loss: 11.4885 - val_accu
racy: 0.6019
Epoch 42/50
14/14 [==============================] - 34s 2s/step - loss: 0.1413 - accuracy: 0.9838 - val_loss: 12.9506 - val_accu
racy: 0.5926
Epoch 43/50
14/14 [==============================] - 34s 2s/step - loss: 0.0742 - accuracy: 0.9884 - val_loss: 13.4545 - val_accu
racy: 0.6389
Epoch 44/50
14/14 [==============================] - 34s 2s/step - loss: 0.0710 - accuracy: 0.9838 - val_loss: 18.5394 - val_accu
racy: 0.3981
Epoch 45/50
14/14 [==============================] - 34s 2s/step - loss: 0.2683 - accuracy: 0.9606 - val_loss: 15.6820 - val_accu
racy: 0.5093
Epoch 46/50
14/14 [==============================] - 34s 2s/step - loss: 0.1978 - accuracy: 0.9653 - val_loss: 13.5089 - val_accu
racy: 0.5833
Epoch 47/50
14/14 [==============================] - 35s 3s/step - loss: 0.0353 - accuracy: 0.9907 - val_loss: 18.4737 - val_accu
racy: 0.6019
Epoch 48/50
14/14 [==============================] - 35s 3s/step - loss: 0.1671 - accuracy: 0.9769 - val_loss: 15.9452 - val_accu
racy: 0.5556
Epoch 49/50
14/14 [==============================] - 34s 2s/step - loss: 0.0635 - accuracy: 0.9907 - val_loss: 12.8523 - val_accu
racy: 0.6296
Epoch 50/50
14/14 [==============================] - 34s 2s/step - loss: 0.1009 - accuracy: 0.9838 - val_loss: 17.2710 - val_accu
racy: 0.6111
WARNING:tensorflow:5 out of the last 9 calls to <function Model.make_predict_function.<locals>.predict_function at 0x
0000020137B0D280> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be
due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python
objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has e
xperimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), plea
se refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/p
ython/tf/function for  more details.
----------------Test accuracy for 5 fold----------------
Confusion Matrix :
 [[24  0  5  1]
 [ 4  4  1  1]
 [ 1  1  8  0]
 [ 8  0  1  1]]
Accuracy    :  0.6166666666666667
Specificity :  0.8058780529755761
Sensitivity :  0.525
-----------------------------End of 5 Fold-------------------------
```

Test Evaluation Results

In [17]:
```
test_accuracy
```

Out[17]:
```
[0.5166666666666667, 0.6166666666666667, 0.65, 0.35, 0.6166666666666667]
```

In [18]:
```
mean_test_accuracy=np.mean(test_accuracy)
mean_test_accuracy
```

Out[18]:
```
0.55
```

In [19]:
```
test_sensitivity
```

Out[19]:
```
[0.39166666666666666, 0.4916666666666667, 0.575, 0.35833333333333334, 0.525]
```

In [20]:
```
mean_test_sensitivity= np.mean(test_sensitivity)
mean_test_sensitivity
```

Out[20]:
```
0.4683333333333334
```

In [21]:
```
test_specificity
```

Out[21]:
```
[0.7276060277483427,
 0.8059116809116809,
 0.8321581196581196,
 0.6559198024715266,
 0.8058780529755761]
```

In [22]:
```
mean_test_specificity= np.mean(test_specificity)
mean_test_specificity
```

Out[22]:
```
0.7654947367530492
```

Training and Validation Evaluation Results

In [23]:
```
train_acc
```

Out[23]:
```
array([0.45833334, 0.54166669, 0.55787039, 0.6412037 , 0.6388889 ,
       0.69907409, 0.71527779, 0.76851851, 0.84027779, 0.87962961,
       0.87037039, 0.81712961, 0.9074074 , 0.93518519, 0.91435188,
0.91666669, 0.92592591, 0.93518519, 0.94444442, 0.9675926 ,
0.9675926 , 0.9699074 , 0.9375    , 0.93518519, 0.94444442,
0.97685188, 0.9699074 , 0.96064812, 0.94444442, 0.9675926 ,
0.97222221, 0.97685188, 0.94907409, 0.97685188, 0.96064812,
0.94675928, 0.97916669, 0.97222221, 0.97685188, 0.97685188,
0.9861111 , 0.95833331, 0.95833331, 0.98842591, 0.96064812,
0.9675926 , 0.9837963 , 0.97685188, 0.99305558, 0.98148149,
0.4074074 , 0.56944442, 0.63194442, 0.62268519, 0.7175926 ,
0.75462961, 0.74768519, 0.8125    , 0.83564812, 0.87731481,
0.7986111 , 0.88425928, 0.93055558, 0.90046299, 0.91203701,
0.91435188, 0.91898149, 0.96064812, 0.9675926 , 0.91435188,
0.94907409, 0.94444442, 0.94675928, 0.96064812, 0.96296299,
0.97916669, 0.94907409, 0.95601851, 0.9675926 , 0.94907409,
0.9375    , 0.98148149, 0.96527779, 0.9837963 , 0.93981481,
0.97453701, 0.9699074 , 0.9675926 , 0.9861111 , 0.97453701,
0.9837963 , 0.9837963 , 0.96296299, 0.9837963 , 0.96296299,
0.98148149, 0.9837963 , 0.9861111 , 0.97916669, 0.99305558,
0.41203704, 0.4699074 , 0.625     , 0.5324074 , 0.67824072,
0.66435188, 0.74074072, 0.77314812, 0.81712961, 0.8287037 ,
0.8587963 , 0.88425928, 0.875     , 0.89583331, 0.9074074 ,
0.87962961, 0.92824072, 0.91435188, 0.96296299, 0.9236111 ,
0.9375    , 0.95601851, 0.9537037 , 0.96527779, 0.95601851,
0.91435188, 0.98842591, 0.9699074 , 0.9675926 , 0.95833331,
0.9513889 , 0.97453701, 0.96296299, 0.97916669, 0.9699074 ,
0.97916669, 0.9675926 , 0.9861111 , 0.98842591, 0.96296299,
0.97685188, 0.97222221, 0.97222221, 0.98842591, 0.95833331,
0.9861111 , 0.98148149, 0.95833331, 0.98842591, 0.99305558,
0.41898149, 0.51157409, 0.57175928, 0.62268519, 0.64814812,
0.72685188, 0.72916669, 0.74768519, 0.8287037 , 0.8425926 ,
0.88657409, 0.9074074 , 0.93287039, 0.89351851, 0.9236111 ,
0.93981481, 0.9236111 , 0.94675928, 0.88425928, 0.93287039,
0.95601851, 0.95601851, 0.96296299, 0.94212961, 0.98148149,
0.94212961, 0.9675926 , 0.9699074 , 0.9537037 , 0.9513889 ,
0.97222221, 0.97453701, 0.97453701, 0.96527779, 0.95833331,
0.97916669, 0.95833331, 0.98148149, 0.96064812, 0.96296299,
0.99537039, 0.97222221, 0.98842591, 0.97685188, 0.99074072,
0.97916669, 0.9837963 , 0.97222221, 0.98842591, 0.9861111 ,
0.42592594, 0.50462961, 0.5787037 , 0.5787037 , 0.64583331,
0.71296299, 0.7013889 , 0.75462961, 0.86342591, 0.81944442,
0.91898149, 0.86574072, 0.9236111 , 0.90972221, 0.9236111 ,
0.9537037 , 0.90277779, 0.9699074 , 0.90972221, 0.93518519,
0.93518519, 0.96064812, 0.96064812, 0.96527779, 0.93518519,
0.93518519, 0.9513889 , 0.96527779, 0.98842591, 0.99074072,
0.96064812, 0.96064812, 0.9699074 , 0.9537037 , 0.9861111 ,
0.99074072, 0.94907409, 0.97685188, 0.98148149, 0.97916669,
0.96527779, 0.9837963 , 0.98842591, 0.9837963 , 0.96064812,
0.96527779, 0.99074072, 0.97685188, 0.99074072, 0.9837963 ])
```

In [24]:
```
mean_train_accuracy=np.mean(train_acc)
mean_train_accuracy
```

Out[24]:
```
0.8978425928354263
```

In [25]:
```
val_acc
```

Out[25]:
```
array([0.2962963 , 0.30555555, 0.37962964, 0.25      , 0.1574074 ,
       0.33333334, 0.5925926 , 0.53703701, 0.28703704, 0.6111111 ,
       0.3425926 , 0.3611111 , 0.44444445, 0.46296296, 0.46296296,
       0.35185185, 0.52777779, 0.44444445, 0.4537037 , 0.23148148,
       0.57407409, 0.52777779, 0.4074074 , 0.5462963 , 0.64814812,
       0.62037039, 0.6574074 , 0.57407409, 0.62037039, 0.56481481,
       0.62037039, 0.52777779, 0.62037039, 0.56481481, 0.53703701,
       0.56481481, 0.5462963 , 0.5925926 , 0.48148149, 0.5925926 ,
       0.49074075, 0.62037039, 0.56481481, 0.62962961, 0.52777779,
       0.5925926 , 0.5925926 , 0.5925926 , 0.56481481, 0.6111111 ,
       0.19444445, 0.5       , 0.43518519, 0.31481481, 0.37037036,
       0.41666666, 0.3425926 , 0.2962963 , 0.39814815, 0.2962963 ,
       0.25925925, 0.5       , 0.49074075, 0.51851851, 0.5462963 ,
       0.48148149, 0.49074075, 0.51851851, 0.31481481, 0.62037039,
       0.43518519, 0.57407409, 0.53703701, 0.50925928, 0.57407409,
       0.53703701, 0.60185188, 0.5925926 , 0.60185188, 0.5925926 ,
       0.51851851, 0.56481481, 0.44444445, 0.41666666, 0.32407406,
       0.53703701, 0.43518519, 0.56481481, 0.57407409, 0.53703701,
       0.49074075, 0.5462963 , 0.57407409, 0.55555558, 0.60185188,
       0.58333331, 0.57407409, 0.5925926 , 0.55555558, 0.5925926 ,
       0.25      , 0.48148149, 0.18518518, 0.18518518, 0.42592594,
       0.37037036, 0.25925925, 0.2962963 , 0.44444445, 0.37037036,
       0.46296296, 0.21296297, 0.43518519, 0.43518519, 0.53703701,
       0.4074074 , 0.26851851, 0.33333334, 0.3888889 , 0.44444445,
       0.58333331, 0.51851851, 0.6111111 , 0.52777779, 0.44444445,
       0.57407409, 0.60185188, 0.51851851, 0.46296296, 0.42592594,
       0.5       , 0.43518519, 0.41666666, 0.44444445, 0.46296296,
       0.37962964, 0.49074075, 0.55555558, 0.49074075, 0.55555558,
       0.60185188, 0.57407409, 0.58333331, 0.60185188, 0.6111111 ,
       0.55555558, 0.37962964, 0.57407409, 0.57407409, 0.6111111 ,
       0.3611111 , 0.2037037 , 0.25      , 0.37962964, 0.41666666,
       0.28703704, 0.57407409, 0.33333334, 0.30555555, 0.2962963 ,
       0.58333331, 0.56481481, 0.5925926 , 0.60185188, 0.49074075,
       0.62037039, 0.5462963 , 0.46296296, 0.5462963 , 0.42592594,
       0.42592594, 0.57407409, 0.48148149, 0.49074075, 0.52777779,
       0.62962961, 0.41666666, 0.3611111 , 0.5462963 , 0.64814812,
       0.62037039, 0.6111111 , 0.62962961, 0.6111111 , 0.58333331,
       0.56481481, 0.58333331, 0.56481481, 0.5925926 , 0.62037039,
       0.62962961, 0.4537037 , 0.52777779, 0.52777779, 0.52777779,
       0.53703701, 0.5       , 0.58333331, 0.56481481, 0.4537037 ,
       0.52777779, 0.52777779, 0.1574074 , 0.2962963 , 0.43518519,
       0.48148149, 0.52777779, 0.46296296, 0.3888889 , 0.49074075,
       0.55555558, 0.47222221, 0.5       , 0.35185185, 0.4537037 ,
       0.57407409, 0.5925926 , 0.51851851, 0.30555555, 0.30555555,
       0.55555558, 0.56481481, 0.43518519, 0.60185188, 0.55555558,
       0.58333331, 0.5925926 , 0.46296296, 0.56481481, 0.51851851,
       0.55555558, 0.3888889 , 0.55555558, 0.42592594, 0.4537037 ,
       0.4537037 , 0.52777779, 0.57407409, 0.52777779, 0.56481481,
       0.60185188, 0.5925926 , 0.6388889 , 0.39814815, 0.50925928,
       0.58333331, 0.60185188, 0.55555558, 0.62962961, 0.6111111 ])
```

In [26]:
```
mean_val_accuracy=np.mean(val_acc)
mean_val_accuracy
```

Out[26]:
```
0.49062963223457334
```

In [27]:
```
train_loss
```

Out[27]:
```
array([1.48859978e+01, 7.04742765e+00, 7.13954687e+00, 3.26615858e+00,
       4.91187525e+00, 2.74668908e+00, 2.65329766e+00, 2.14300656e+00,
       1.38960099e+00, 1.29897487e+00, 9.35277522e-01, 1.31745064e+00,
       4.85455334e-01, 4.23509032e-01, 4.86319810e-01, 4.38042879e-01,
       4.66184169e-01, 4.90887254e-01, 6.41161740e-01, 1.74766377e-01,
       2.57000327e-01, 1.69235244e-01, 5.19961655e-01, 5.18275440e-01,
       3.80493760e-01, 1.39512450e-01, 1.93364114e-01, 3.01398695e-01,
       2.96037763e-01, 4.00338411e-01, 2.64167905e-01, 1.82599902e-01,
       2.91099817e-01, 3.07638615e-01, 2.86613107e-01, 6.00891352e-01,
       7.21284747e-02, 1.69293419e-01, 1.34037510e-01, 1.37226701e-01,
       1.26281485e-01, 5.29202282e-01, 2.73458958e-01, 1.33905783e-01,
       2.84882188e-01, 3.38376790e-01, 1.80160508e-01, 2.08130136e-01,
       7.84452334e-02, 1.41649365e-01, 1.54807987e+01, 7.25753546e+00,
       4.96772480e+00, 5.32112837e+00, 2.34611011e+00, 2.13071632e+00,
       3.39239383e+00, 1.33588803e+00, 7.74274349e-01, 7.25334942e-01,
       1.65536475e+00, 6.84605181e-01, 4.43887353e-01, 7.27934241e-01,
       6.46314979e-01, 6.22846723e-01, 6.95517659e-01, 2.40133226e-01,
       1.89909473e-01, 5.64030766e-01, 3.30894142e-01, 2.85961121e-01,
       5.93639612e-01, 2.26651281e-01, 2.46011764e-01, 1.75064296e-01,
       7.17682600e-01, 3.42307866e-01, 1.89056203e-01, 3.37093204e-01,
       4.04514253e-01, 9.97474492e-02, 1.82234541e-01, 8.95176232e-02,
       6.36588812e-01, 1.39627859e-01, 2.16375098e-01, 3.30170780e-01,
       8.20210427e-02, 1.49377421e-01, 1.76813722e-01, 1.90761209e-01,
       1.90043896e-01, 3.76738831e-02, 2.20002115e-01, 5.98452874e-02,
       9.08400118e-02, 1.18905194e-01, 1.71879560e-01, 2.68180128e-02,
       1.82884541e+01, 8.49352360e+00, 4.87486887e+00, 6.28171587e+00,
       3.26819038e+00, 4.09774351e+00, 2.02038145e+00, 1.78624439e+00,
       1.35905981e+00, 1.18741381e+00, 1.45854008e+00, 6.89337969e-01,
       8.92998040e-01, 5.91831684e-01, 5.03535032e-01, 7.31126666e-01,
       4.90795583e-01, 5.77540100e-01, 7.79042542e-01, 4.21679258e-01,
       5.04003286e-01, 2.90027142e-01, 2.62509376e-01, 2.76915908e-01,
       2.63967276e-01, 1.26812065e+00, 5.23396470e-02, 3.61801982e-01,
       1.84317052e-01, 3.36810529e-01, 5.44393778e-01, 1.85401157e-01,
       1.96147561e-01, 2.02178687e-01, 1.81016237e-01, 1.28030807e-01,
       3.37217003e-01, 3.61936130e-02, 1.20347828e-01, 2.78631955e-01,
       1.89363509e-01, 2.13081822e-01, 2.73969591e-01, 5.50072864e-02,
       2.20493883e-01, 5.94365895e-02, 6.26017079e-02, 3.33609402e-01,
       6.75709918e-02, 1.21852018e-01, 1.76823330e+01, 8.01148319e+00,
       5.98482609e+00, 4.90558195e+00, 4.60781002e+00, 1.98446512e+00,
       2.49319100e+00, 2.02873874e+00, 1.20051098e+00, 8.89535069e-01,
       7.21348941e-01, 1.25382590e+00, 4.34716552e-01, 6.56889617e-01,
       4.26324666e-01, 3.75432372e-01, 6.40710711e-01, 4.17907923e-01,
       9.70365107e-01, 4.71083015e-01, 2.88479567e-01, 1.25194937e-01,
       2.17559308e-01, 3.91157746e-01, 2.38995552e-01, 3.32989901e-01,
       2.25520954e-01, 1.71489447e-01, 3.33664507e-01, 2.54938871e-01,
       1.27974436e-01, 1.31266162e-01, 1.24774501e-01, 1.60628691e-01,
       3.08604002e-01, 1.01914212e-01, 2.55537063e-01, 8.15532133e-02,
       4.10412937e-01, 2.32607663e-01, 1.12919752e-02, 1.65526360e-01,
       8.22251812e-02, 9.59703103e-02, 4.73319292e-02, 1.57143861e-01,
       1.10556617e-01, 2.33007208e-01, 3.84275839e-02, 9.53883976e-02,
       1.64354954e+01, 7.54237938e+00, 5.13609743e+00, 5.86207581e+00,
       3.18880916e+00, 2.19675994e+00, 3.76073027e+00, 1.88379729e+00,
       7.08827674e-01, 1.91635740e+00, 3.95489365e-01, 9.23473001e-01,
       4.25988257e-01, 5.28674960e-01, 5.75568616e-01, 9.80410397e-01,
       6.29643559e-01, 1.43865764e-01, 6.90543294e-01, 6.32255912e-01,
       4.04937297e-01, 1.72960684e-01, 3.36907566e-01, 4.04046565e-01,
       4.57929671e-01, 6.71902895e-01, 2.10089073e-01, 2.89691120e-01,
       3.93495895e-02, 3.53777818e-02, 3.24405640e-01, 3.09244037e-01,
       2.18345731e-01, 2.95967102e-01, 6.82382956e-02, 6.88140765e-02,
       3.43545735e-01, 1.14859864e-01, 1.96595222e-01, 1.55734360e-01,
       2.95766145e-01, 1.41314790e-01, 7.42241666e-02, 7.09837079e-02,
       2.68266559e-01, 1.97793588e-01, 3.53235044e-02, 1.67108402e-01,
       6.34959564e-02, 1.00949973e-01])
```

In [28]:
```
mean_train_loss=np.mean(train_loss)
mean_train_loss
```

Out[28]:
```
1.2308800502121449
```

In [29]:
```
val_loss
```

Out[29]:
```
array([ 89.66611481,  22.09983826,  10.17586422,  44.8885994 ,
        21.22432518,   7.66131687,   5.64215183,   3.03474164,
         8.42742729,   3.46080923,   8.02373505,   6.46243715,
         4.56035709,   4.43408298,   9.26592827,  11.20942402,
         8.16215706,  12.13919735,   8.53639889,  13.00899601,
         5.80073595,   5.3161478 ,  11.29268646,   9.1288414 ,
         8.1673317 ,   6.76663113,   7.6253562 ,   9.06472778,
         9.12248421,   7.62374163,   6.21712971,   8.99226189,
         5.95892572,  10.82538319,  22.35163498,  13.13454723,
        13.15687943,   9.37051296,   9.58472061,   7.7275548 ,
        10.06571484,  11.40365791,   7.41928101,   9.26370811,
        12.07525158,   7.61726046,   9.75984287,  11.20426083,
        10.02727032,   9.93282986, 101.04224396,  25.88222885,
        36.09226608,  16.20354462,  11.30362034,  10.73334789,
        12.22359848,  13.2954216 ,   6.71817827,  10.91052151,
        20.38394547,   6.15647221,   6.21141577,  11.91195774,
         6.08360863,   9.45283318,   7.99971676,   7.08172083,
        15.15541935,   9.65091324,   7.43398714,   9.95938969,
         6.78519917,   8.68740749,   7.87415981,   8.06662369,
         9.51844692,  15.99774361,  12.81514835,  12.25922298,
         9.91177559,  12.14988422,  13.30214882,  17.96403885,
        21.33241272,  23.80174065,  22.17542648,  29.86895752,
        23.50940704,  56.6337738 ,  49.65166855,  44.11458969,
        46.5102005 ,  71.98677826,  57.3072052 ,  45.09919739,
        28.86948586,  10.30592823,  36.25561142,  28.11395454,
        55.86030579,  25.69961166,  36.75431442,  18.68954849,
         6.27147102,   7.8781991 ,  12.06573486,  10.95243931,
         7.08436394,   5.91762972,   9.98108959,  12.80846405,
         5.98620415,   5.7060113 ,   6.78061247,   8.79601669,
         9.18163204,  14.62390995,   8.25587273,   6.44002867,
         7.75073338,   5.24204779,   5.68341684,   5.65307426,
         8.65698528,   6.97773361,   6.29120779,   9.27094936,
         9.19501209,  11.82020664,   9.30251598,  10.22557926,
        11.31398296,   9.69942665,  10.08496571,  13.07944393,
         8.42576122,   8.17350292,   8.9804554 ,   9.76876259,
         9.04224968,  17.81778526,   9.44031239,  11.39252377,
        13.06479073,  13.18155003,  15.88471317,  11.00124741,
        10.2443924 ,  12.87900639,  88.98149109,  54.84194565,
        35.09395981,  21.26029968,   6.20234776,  21.83916855,
        10.89391422,   6.936975  ,   6.98031187,   5.52201414,
         5.3876791 ,   6.65837383,   7.13228464,   7.75519991,
         7.98684502,   5.33093262,   7.22934961,   5.28315401,
         5.75167465,   6.47032356,   4.67685318,   6.18563557,
         9.54906559,   7.53847599,   8.42907906,   6.27483273,
         7.3196888 ,  11.31374931,   8.46966076,   6.86419678,
         6.80712986,   7.28560781,   6.97031212,  16.35381126,
         6.33203077,   6.9896965 ,   6.84285212,   7.87982225,
         9.78041553,   9.03537178,   7.54497957,   8.63079453,
         5.38333321,   7.27706957,   7.10836792,   6.77295113,
         9.34701729,   7.48261976,   8.23129177,   9.49849033,
        61.956707  ,  13.84055996,  43.1405983 ,  13.8490715 ,
        18.54131699,  10.08979034,   4.9406476 ,   6.48847342,
         6.23208952,   3.83363819,   3.69018149,   6.63373137,
         5.97931576,  11.98450279,  11.2626543 ,   6.12480688,
         5.56270361,   7.52238703,  17.24069977,  11.43555069,
         5.61269855,   5.21371889,  11.60570908,  10.24027538,
        10.94992065,   7.91949797,   9.69431305,   8.2393713 ,
         7.3632865 ,   8.11259365,   7.53622961,  12.06420231,
         7.66099882,  12.73732471,  11.53381538,  10.38310623,
        16.05274963,  16.99931335,  15.38810921,  15.91551113,
        11.4885416 ,  12.95060921,  13.45446205,  18.53936577,
        15.68201447,  13.50891113,  18.47370338,  15.94522095,
        12.85233974,  17.27095413])
```

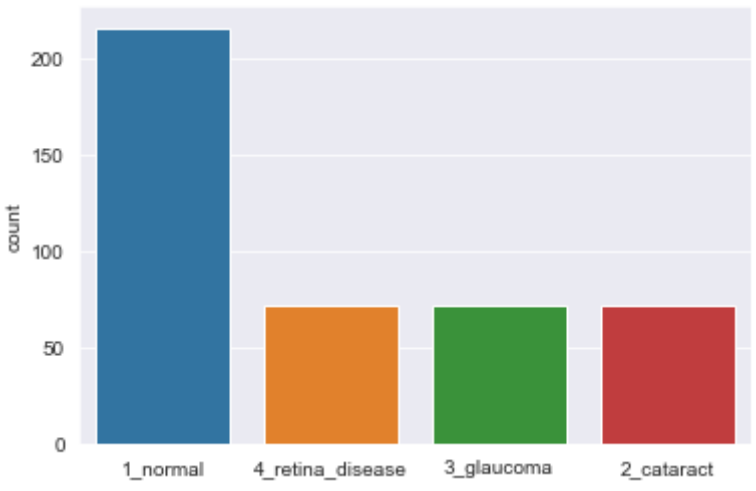
In [30]:
```
mean_val_loss=np.mean(val_loss)
mean_val_loss
```

Out[30]:
```
13.798260963439942
```

Plot to Visualize the Number of Images in Each Label of Trainig Dataset

In [31]:
```
l = []
for i in train:
    if(i[1] == 0):
        l.append("1_normal")

    elif (i[1] == 1):
        l.append("2_cataract")

    elif (i[1] == 2):
        l.append("3_glaucoma")

    else :
        l.append("4_retina_disease")


sns.set_style('darkgrid')
sns.countplot(l)
```

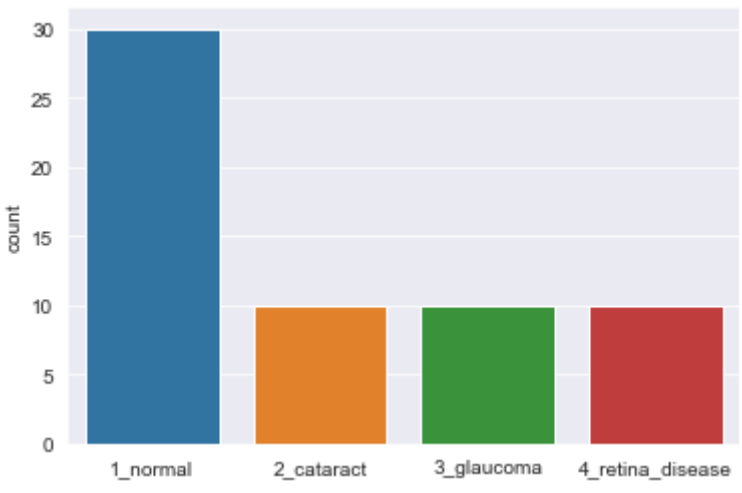
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x2013f31cf10>



Plot to Visualize the Number of Images in Each Label of Test Dataset.

In [32]:
```
l = []
for i in test:
    if(i[1] == 0):
        l.append("1_normal")

    elif (i[1] == 1):
        l.append("2_cataract")

    elif (i[1] == 2):
        l.append("3_glaucoma")

    else :
        l.append("4_retina_disease")


sns.set_style('darkgrid')
sns.countplot(l)
```

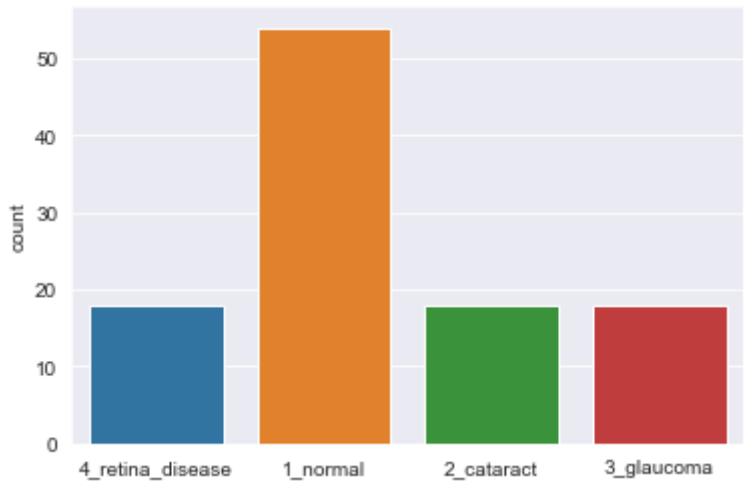
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x2012fc78820>



Plot to Visualize the Number of Images in Each Label of Validation Dataset.

In [33]:
```
l = []
for i in validation:
    if(i[1] == 0):
        l.append("1_normal")

    elif (i[1] == 1):
        l.append("2_cataract")

    elif (i[1] == 2):
        l.append("3_glaucoma")

    else :
        l.append("4_retina_disease")


sns.set_style('darkgrid')
sns.countplot(l)
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x2013118be20>

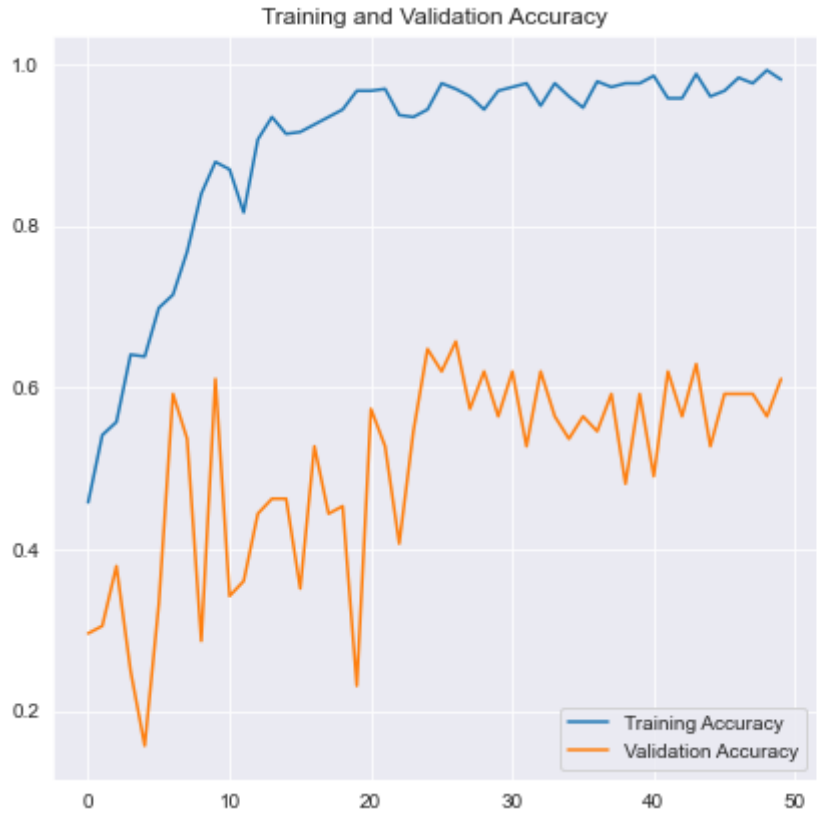


Training,Validation Accuracy and Loss Plot for 50 Epochs

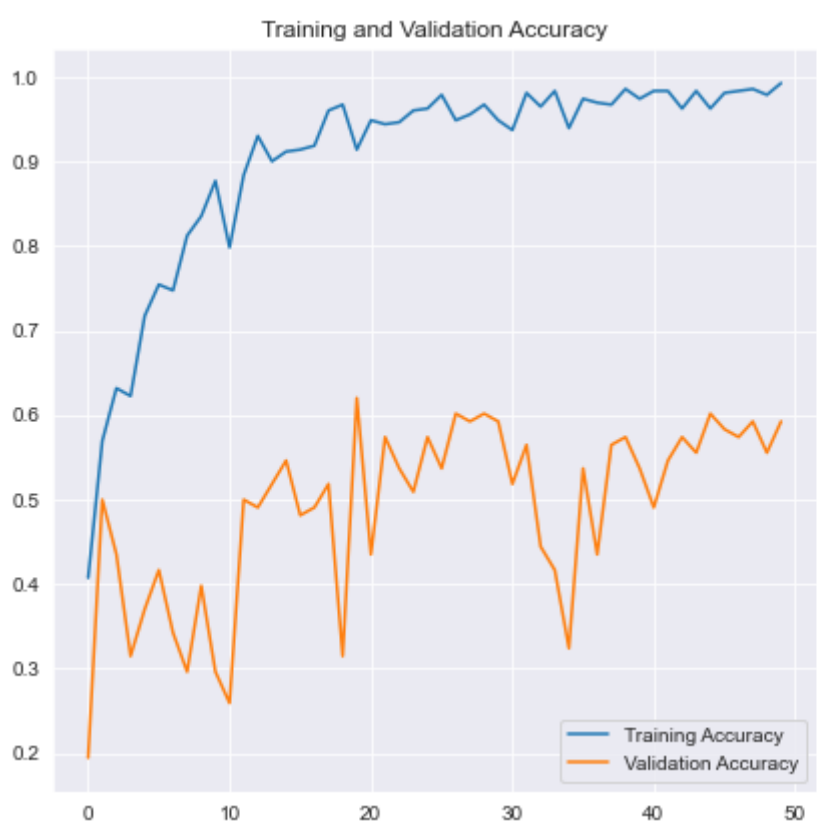
In [34]:
```
def plot_print(i,j):
    epochs_range = range(50)

    plt.figure(figsize=(15, 15))
    plt.subplot(2, 2, 1)
    plt.plot(epochs_range, train_acc[i:j], label='Training Accuracy')
    plt.plot(epochs_range, val_acc[i:j], label='Validation Accuracy')
    plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy')

    plt.subplot(2, 2, 2)
    plt.plot(epochs_range, train_loss[i:j], label='Training Loss')
    plt.plot(epochs_range, val_loss[i:j], label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')

    return plt.show()
```

In [35]:
```
k=1
j=0
for i in range(0,250,50):
    j +=50
    print('Plot for ',k,'cross validation accuracy and loss for Training and Validation phase')
    k +=1
    plot_print(i,j)
```

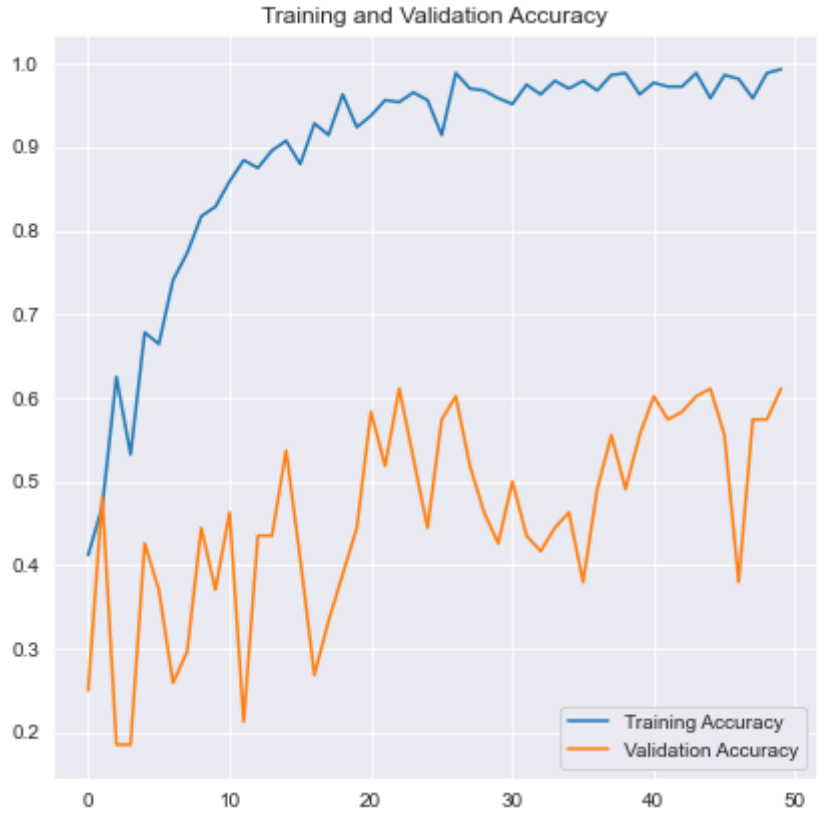
Plot for 1 cross validation accuracy and loss for Training and Validation phase



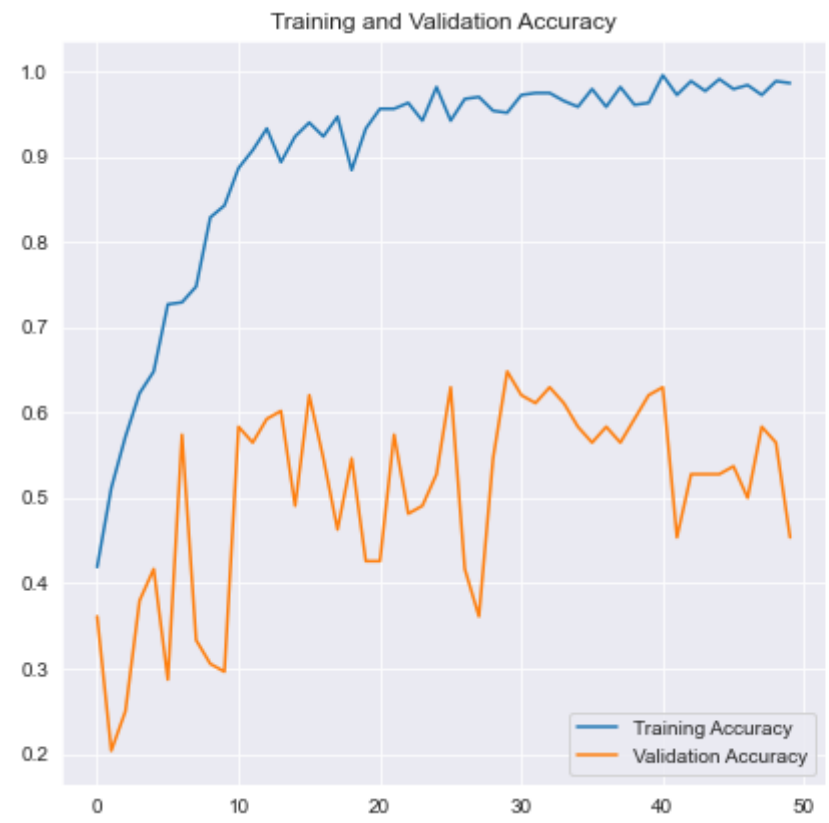
Plot for 2 cross validation accuracy and loss for Training and Validation phase



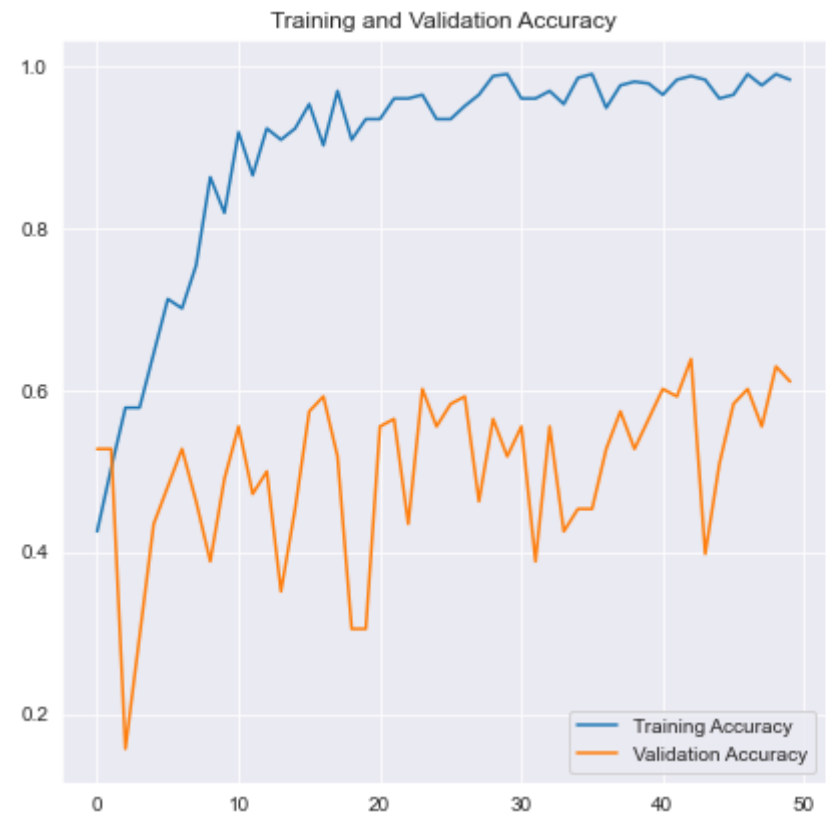
Plot for 3 cross validation accuracy and loss for Training and Validation phase



Plot for 4 cross validation accuracy and loss for Training and Validation phase



Plot for 5 cross validation accuracy and loss for Training and Validation phase



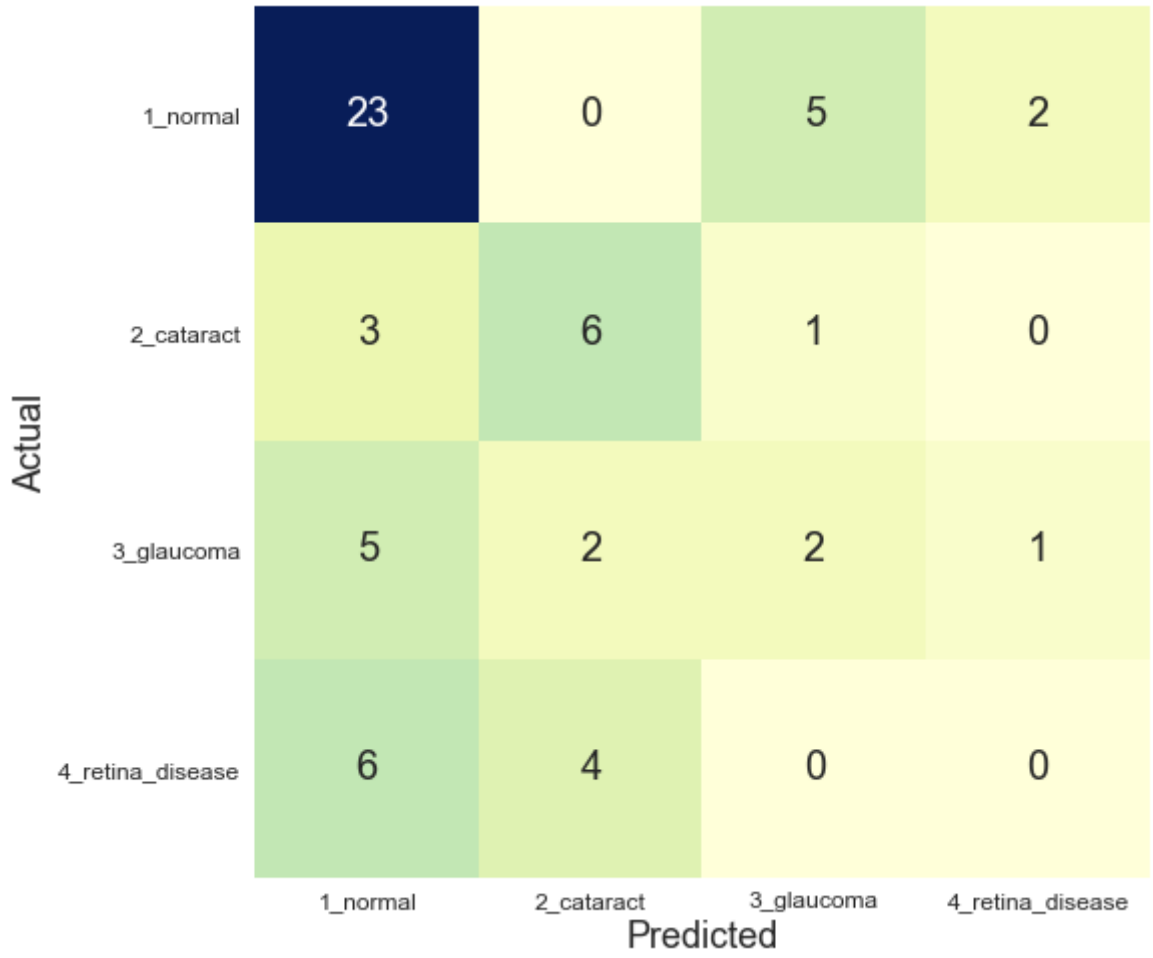
Visualizing Confusion Matrix for Each Fold

In [36]:
```
CM= np.array(CM)
CM.resize(5,4,4)
```

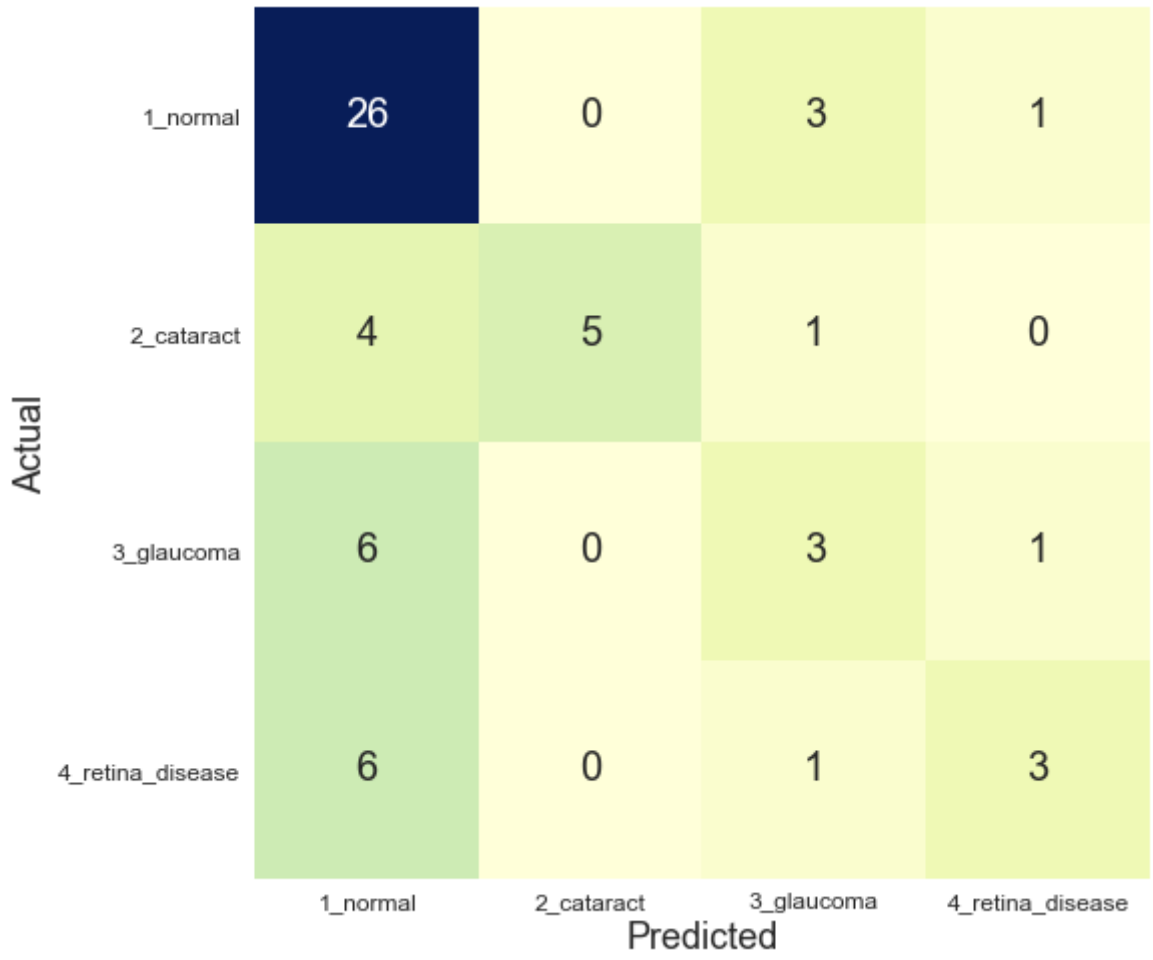
In [37]:
```
def confusionmatrix_vis(i):

        yticklabels=['1_normal', '2_cataract','3_glaucoma','4_retina_disease']
        xticklabels=['1_normal', '2_cataract','3_glaucoma','4_retina_disease']
        plt.figure(figsize=(8, 8))
        hm =sns.heatmap(CM[i], annot=True,annot_kws={"size": 20}, cbar=False,cmap="YlGnBu",yticklabels=yticklabels,xti
cklabels=xticklabels)

        hm.set_xticklabels(hm.get_xticklabels(), rotation=0, fontsize = 12, )
        hm.set_yticklabels(hm.get_yticklabels(), rotation=0, fontsize = 12)

        plt.ylabel("Actual", fontsize = 18)
        plt.xlabel("Predicted",fontsize = 18)

        return plt.show()
```

In [38]:
```
k=1
for i in range(5):
    print('Confusion Matrix for ',k,'Cross Validation Test phase')
    k +=1
    confusionmatrix_vis(i)
```

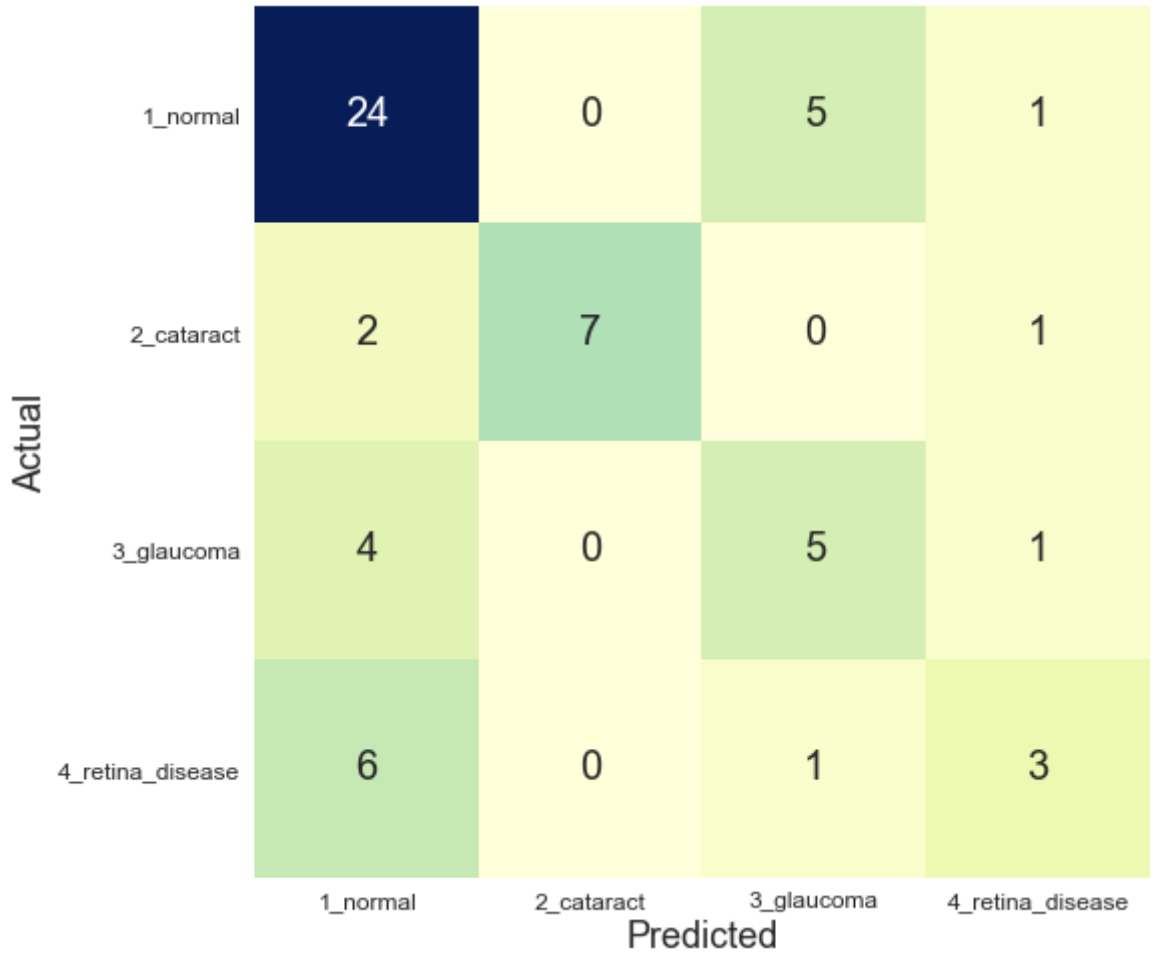
Confusion Matrix for 1 Cross Validation Test phase



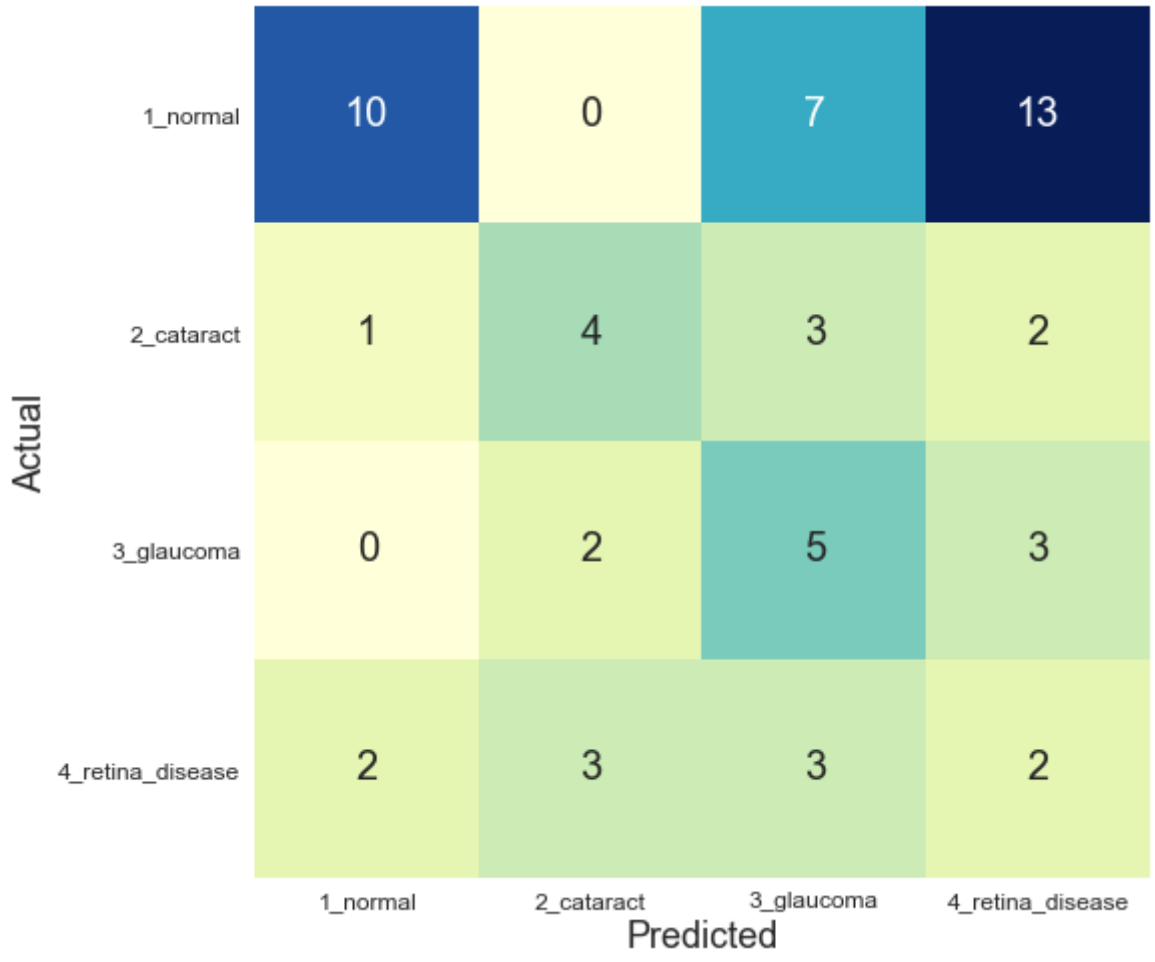
Confusion Matrix for 2 Cross Validation Test phase



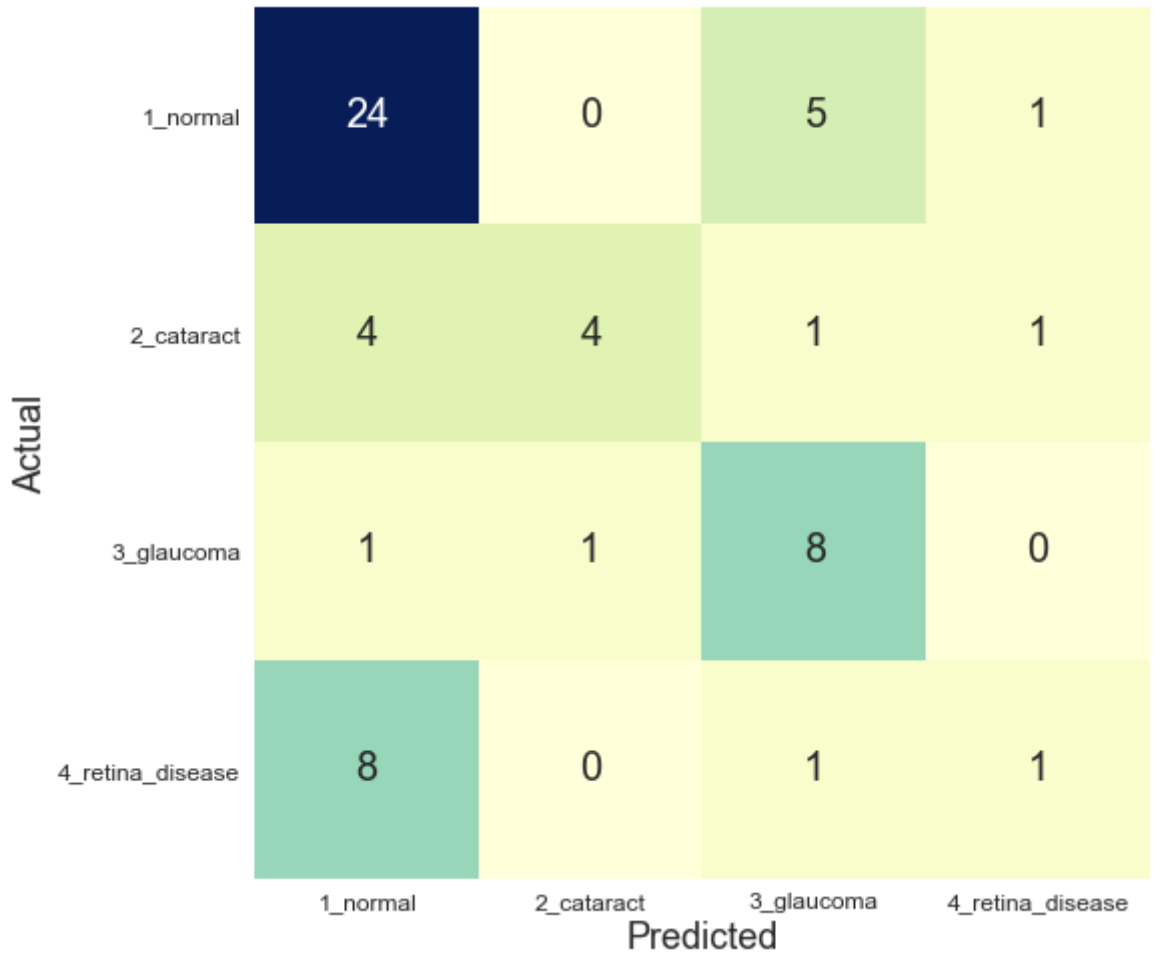
Confusion Matrix for 3 Cross Validation Test phase



Confusion Matrix for 4 Cross Validation Test phase



Confusion Matrix for 5 Cross Validation Test phase

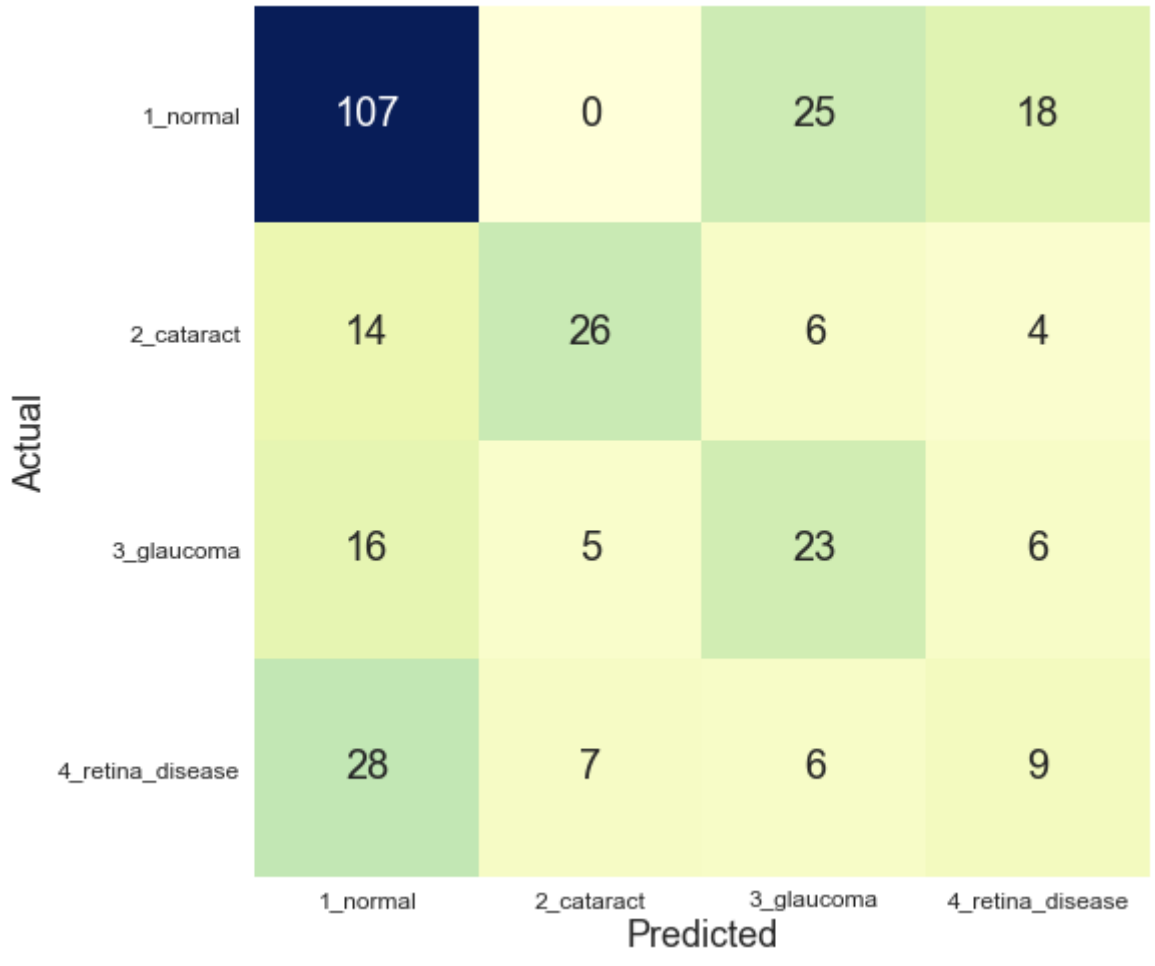


Visualizing Summarized Confusion Matrix of all 5 folds

In [39]:
```
CM_sum = CM[0]+CM[1]+CM[2]+CM[3]+CM[4]
CM_sum
```

Out[39]:
```
array([[107,   0,  25,  18],
       [ 14,  26,   6,   4],
       [ 16,   5,  23,   6],
       [ 28,   7,   6,   9]], dtype=int64)
```

In [47]:
```
yticklabels=['1_normal', '2_cataract','3_glaucoma','4_retina_disease']
xticklabels=['1_normal', '2_cataract','3_glaucoma','4_retina_disease']
plt.figure(figsize=(8, 8))
hm =sns.heatmap(CM_sum, annot=True,annot_kws={"size": 20},fmt='g', cbar=False,cmap="YlGnBu",yticklabels=yticklabels,xt
icklabels=xticklabels)

hm.set_xticklabels(hm.get_xticklabels(), rotation=0, fontsize = 12, )
hm.set_yticklabels(hm.get_yticklabels(), rotation=0, fontsize = 12)

plt.ylabel("Actual", fontsize = 18)
plt.xlabel("Predicted",fontsize = 18)

plt.show()
```



Reconfirming the values of Accuracy,Sensitivity and Specificity

In [48]:
```
    sensitivity_1_normal = (CM_sum[0,0])/(CM_sum[0,0]+CM_sum[0,1]+CM_sum[0,2]+CM_sum[0,3])
    #print('Sensitivity_1_normal          : ', sensitivity_1_normal )

    sensitivity_2_cataract = (CM_sum[1,1])/(CM_sum[1,0]+CM_sum[1,1]+CM_sum[1,2]+CM_sum[1,3])
    #print('Sensitivity_2_cataract        : ', sensitivity_2_cataract )


    sensitivity_3_glaucoma = (CM_sum[2,2])/(CM_sum[2,0]+CM_sum[2,1]+CM_sum[2,2]+CM_sum[2,3])
    #print('Sensitivity_3_glaucoma        : ', sensitivity_3_glaucoma )

    sensitivity_4_retina_disease = (CM_sum[3,3])/(CM_sum[3,0]+CM_sum[3,1]+CM_sum[3,2]+CM_sum[3,3])
    #print('Sensitivity_4_retina_disease  : ', sensitivity_4_retina_disease )

    specificity_1_normal = (CM_sum[1,1]+CM_sum[1,2]+CM_sum[1,3]+CM_sum[2,1]+CM_sum[2,2]+CM_sum[2,3]+CM_sum[3,1]+CM_sum
[3,2]+CM_sum[3,3])/(CM_sum[1,0]+CM_sum[2,0]+CM_sum[3,0]+CM_sum[1,1]+CM_sum[1,2]+CM_sum[1,3]+CM_sum[2,1]+CM_sum[2,2]+CM
_sum[2,3]+CM_sum[3,1]+CM_sum[3,2]+CM_sum[3,3])
    #print('Specificity : ', specificity_1_normal)

    specificity_2_cataract = (CM_sum[0,0]+CM_sum[0,2]+CM_sum[0,3]+CM_sum[2,0]+CM_sum[2,2]+CM_sum[2,3]+CM_sum[3,0]+CM_s
um[3,2]+CM_sum[3,3])/(CM_sum[0,1]+CM_sum[2,1]+CM_sum[3,1]+CM_sum[0,0]+CM_sum[0,2]+CM_sum[0,3]+CM_sum[2,0]+CM_sum[2,2]+
CM_sum[2,3]+CM_sum[3,0]+CM_sum[3,2]+CM_sum[3,3])
    #print('Specificity : ', specificity_2_cataract)

    specificity_3_glaucoma = (CM_sum[0,0]+CM_sum[0,1]+CM_sum[0,3]+CM_sum[1,0]+CM_sum[1,1]+CM_sum[1,3]+CM_sum[3,0]+CM_s
um[3,1]+CM_sum[3,3])/(CM_sum[0,2]+CM_sum[1,2]+CM_sum[3,2]+CM_sum[0,0]+CM_sum[0,1]+CM_sum[0,3]+CM_sum[1,0]+CM_sum[1,1]+
CM_sum[1,3]+CM_sum[3,0]+CM_sum[3,1]+CM_sum[3,3])
    #print('Specificity : ', specificity_3_glaucoma)

    specificity_4_retina_disease= (CM_sum[0,0]+CM_sum[0,1]+CM_sum[0,2]+CM_sum[1,0]+CM_sum[1,1]+CM_sum[1,2]+CM_sum[2,0]
+CM_sum[2,1]+CM_sum[2,2])/(CM_sum[0,3]+CM_sum[1,3]+CM_sum[2,3]+CM_sum[0,0]+CM_sum[0,1]+CM_sum[0,2]+CM_sum[1,0]+CM_sum[
1,1]+CM_sum[1,2]+CM_sum[2,0]+CM_sum[2,1]+CM_sum[2,2])
    #print('Specificity : ', specificity_4_retina_disease)

    Sensitivity= (sensitivity_1_normal + sensitivity_2_cataract + sensitivity_3_glaucoma + sensitivity_4_retina_diseas
e)/4
    #print(Sensitivity)

    Specificity= (specificity_1_normal + specificity_2_cataract + specificity_3_glaucoma + specificity_4_retina_diseas
e)/4
    #print(Specificity)

    total1=sum(sum(CM_sum))
    test_accuracy=(CM_sum[0,0]+CM_sum[1,1]+CM_sum[2,2]+CM_sum[3,3])/total1

    print ('Accuracy    : ', test_accuracy)
    print ('Specificity : ', Specificity)
    print ('Sensitivity : ', Sensitivity)
```

```
Accuracy    :  0.55
Specificity :  0.765412994416625
Sensitivity :  0.4683333333333333
```

Model Summary

In [45]:
```
model_build_compile(k)
```

```
model building and compiling for fold 7
```

Out[45]:
```
<tensorflow.python.keras.engine.functional.Functional at 0x2013c3c9d00>
```

In [46]:
```
model.summary()
```

Model: "model_4"

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[(None, 224, 224, 3)	0	
conv2d_376 (Conv2D)	(None, 111, 111, 32)	864	input_5[0][0]
batch_normalization_388 (BatchN	(None, 111, 111, 32)	96	conv2d_376[0][0]
activation_376 (Activation)	(None, 111, 111, 32)	0	batch_normalization_388[0][0]
conv2d_377 (Conv2D)	(None, 109, 109, 32)	9216	activation_376[0][0]
batch_normalization_389 (BatchN	(None, 109, 109, 32)	96	conv2d_377[0][0]
activation_377 (Activation)	(None, 109, 109, 32)	0	batch_normalization_389[0][0]
conv2d_378 (Conv2D)	(None, 109, 109, 64)	18432	activation_377[0][0]
batch_normalization_390 (BatchN	(None, 109, 109, 64)	192	conv2d_378[0][0]
activation_378 (Activation)	(None, 109, 109, 64)	0	batch_normalization_390[0][0]
max_pooling2d_16 (MaxPooling2D)	(None, 54, 54, 64)	0	activation_378[0][0]
conv2d_379 (Conv2D)	(None, 54, 54, 80)	5120	max_pooling2d_16[0][0]
batch_normalization_391 (BatchN	(None, 54, 54, 80)	240	conv2d_379[0][0]
activation_379 (Activation)	(None, 54, 54, 80)	0	batch_normalization_391[0][0]
conv2d_380 (Conv2D)	(None, 52, 52, 192)	138240	activation_379[0][0]
batch_normalization_392 (BatchN	(None, 52, 52, 192)	576	conv2d_380[0][0]
activation_380 (Activation)	(None, 52, 52, 192)	0	batch_normalization_392[0][0]
max_pooling2d_17 (MaxPooling2D)	(None, 25, 25, 192)	0	activation_380[0][0]
conv2d_384 (Conv2D)	(None, 25, 25, 64)	12288	max_pooling2d_17[0][0]
batch_normalization_396 (BatchN	(None, 25, 25, 64)	192	conv2d_384[0][0]
activation_384 (Activation)	(None, 25, 25, 64)	0	batch_normalization_396[0][0]
conv2d_382 (Conv2D)	(None, 25, 25, 48)	9216	max_pooling2d_17[0][0]
conv2d_385 (Conv2D)	(None, 25, 25, 96)	55296	activation_384[0][0]
batch_normalization_394 (BatchN	(None, 25, 25, 48)	144	conv2d_382[0][0]
batch_normalization_397 (BatchN	(None, 25, 25, 96)	288	conv2d_385[0][0]
activation_382 (Activation)	(None, 25, 25, 48)	0	batch_normalization_394[0][0]
activation_385 (Activation)	(None, 25, 25, 96)	0	batch_normalization_397[0][0]
average_pooling2d_36 (AveragePo	(None, 25, 25, 192)	0	max_pooling2d_17[0][0]
conv2d_381 (Conv2D)	(None, 25, 25, 64)	12288	max_pooling2d_17[0][0]
conv2d_383 (Conv2D)	(None, 25, 25, 64)	76800	activation_382[0][0]
conv2d_386 (Conv2D)	(None, 25, 25, 96)	82944	activation_385[0][0]
conv2d_387 (Conv2D)	(None, 25, 25, 32)	6144	average_pooling2d_36[0][0]
batch_normalization_393 (BatchN	(None, 25, 25, 64)	192	conv2d_381[0][0]
batch_normalization_395 (BatchN	(None, 25, 25, 64)	192	conv2d_383[0][0]
batch_normalization_398 (BatchN	(None, 25, 25, 96)	288	conv2d_386[0][0]
batch_normalization_399 (BatchN	(None, 25, 25, 32)	96	conv2d_387[0][0]
activation_381 (Activation)	(None, 25, 25, 64)	0	batch_normalization_393[0][0]
activation_383 (Activation)	(None, 25, 25, 64)	0	batch_normalization_395[0][0]
activation_386 (Activation)	(None, 25, 25, 96)	0	batch_normalization_398[0][0]
activation_387 (Activation)	(None, 25, 25, 32)	0	batch_normalization_399[0][0]
mixed0 (Concatenate)	(None, 25, 25, 256)	0	activation_381[0][0] activation_383[0][0] activation_386[0][0] activation_387[0][0]

conv2d_391 (Conv2D)	(None, 25, 25, 64)	16384	mixed0[0][0]
batch_normalization_403 (BatchN	(None, 25, 25, 64)	192	conv2d_391[0][0]
activation_391 (Activation)	(None, 25, 25, 64)	0	batch_normalization_403[0][0]
conv2d_389 (Conv2D)	(None, 25, 25, 48)	12288	mixed0[0][0]
conv2d_392 (Conv2D)	(None, 25, 25, 96)	55296	activation_391[0][0]
batch_normalization_401 (BatchN	(None, 25, 25, 48)	144	conv2d_389[0][0]
batch_normalization_404 (BatchN	(None, 25, 25, 96)	288	conv2d_392[0][0]
activation_389 (Activation)	(None, 25, 25, 48)	0	batch_normalization_401[0][0]
activation_392 (Activation)	(None, 25, 25, 96)	0	batch_normalization_404[0][0]
average_pooling2d_37 (AveragePo	(None, 25, 25, 256)	0	mixed0[0][0]
conv2d_388 (Conv2D)	(None, 25, 25, 64)	16384	mixed0[0][0]
conv2d_390 (Conv2D)	(None, 25, 25, 64)	76800	activation_389[0][0]
conv2d_393 (Conv2D)	(None, 25, 25, 96)	82944	activation_392[0][0]
conv2d_394 (Conv2D)	(None, 25, 25, 64)	16384	average_pooling2d_37[0][0]
batch_normalization_400 (BatchN	(None, 25, 25, 64)	192	conv2d_388[0][0]
batch_normalization_402 (BatchN	(None, 25, 25, 64)	192	conv2d_390[0][0]
batch_normalization_405 (BatchN	(None, 25, 25, 96)	288	conv2d_393[0][0]
batch_normalization_406 (BatchN	(None, 25, 25, 64)	192	conv2d_394[0][0]
activation_388 (Activation)	(None, 25, 25, 64)	0	batch_normalization_400[0][0]
activation_390 (Activation)	(None, 25, 25, 64)	0	batch_normalization_402[0][0]
activation_393 (Activation)	(None, 25, 25, 96)	0	batch_normalization_405[0][0]
activation_394 (Activation)	(None, 25, 25, 64)	0	batch_normalization_406[0][0]
mixed1 (Concatenate)	(None, 25, 25, 288)	0	activation_388[0][0] activation_390[0][0] activation_393[0][0] activation_394[0][0]
conv2d_398 (Conv2D)	(None, 25, 25, 64)	18432	mixed1[0][0]
batch_normalization_410 (BatchN	(None, 25, 25, 64)	192	conv2d_398[0][0]
activation_398 (Activation)	(None, 25, 25, 64)	0	batch_normalization_410[0][0]
conv2d_396 (Conv2D)	(None, 25, 25, 48)	13824	mixed1[0][0]
conv2d_399 (Conv2D)	(None, 25, 25, 96)	55296	activation_398[0][0]
batch_normalization_408 (BatchN	(None, 25, 25, 48)	144	conv2d_396[0][0]
batch_normalization_411 (BatchN	(None, 25, 25, 96)	288	conv2d_399[0][0]
activation_396 (Activation)	(None, 25, 25, 48)	0	batch_normalization_408[0][0]
activation_399 (Activation)	(None, 25, 25, 96)	0	batch_normalization_411[0][0]
average_pooling2d_38 (AveragePo	(None, 25, 25, 288)	0	mixed1[0][0]
conv2d_395 (Conv2D)	(None, 25, 25, 64)	18432	mixed1[0][0]
conv2d_397 (Conv2D)	(None, 25, 25, 64)	76800	activation_396[0][0]
conv2d_400 (Conv2D)	(None, 25, 25, 96)	82944	activation_399[0][0]
conv2d_401 (Conv2D)	(None, 25, 25, 64)	18432	average_pooling2d_38[0][0]
batch_normalization_407 (BatchN	(None, 25, 25, 64)	192	conv2d_395[0][0]
batch_normalization_409 (BatchN	(None, 25, 25, 64)	192	conv2d_397[0][0]
batch_normalization_412 (BatchN	(None, 25, 25, 96)	288	conv2d_400[0][0]
batch_normalization_413 (BatchN	(None, 25, 25, 64)	192	conv2d_401[0][0]
activation_395 (Activation)	(None, 25, 25, 64)	0	batch_normalization_407[0][0]

activation_397 (Activation)	(None, 25, 25, 64)	0	batch_normalization_409[0][0]
activation_400 (Activation)	(None, 25, 25, 96)	0	batch_normalization_412[0][0]
activation_401 (Activation)	(None, 25, 25, 64)	0	batch_normalization_413[0][0]
mixed2 (Concatenate)	(None, 25, 25, 288)	0	activation_395[0][0] activation_397[0][0] activation_400[0][0] activation_401[0][0]
conv2d_403 (Conv2D)	(None, 25, 25, 64)	18432	mixed2[0][0]
batch_normalization_415 (BatchN	(None, 25, 25, 64)	192	conv2d_403[0][0]
activation_403 (Activation)	(None, 25, 25, 64)	0	batch_normalization_415[0][0]
conv2d_404 (Conv2D)	(None, 25, 25, 96)	55296	activation_403[0][0]
batch_normalization_416 (BatchN	(None, 25, 25, 96)	288	conv2d_404[0][0]
activation_404 (Activation)	(None, 25, 25, 96)	0	batch_normalization_416[0][0]
conv2d_402 (Conv2D)	(None, 12, 12, 384)	995328	mixed2[0][0]
conv2d_405 (Conv2D)	(None, 12, 12, 96)	82944	activation_404[0][0]
batch_normalization_414 (BatchN	(None, 12, 12, 384)	1152	conv2d_402[0][0]
batch_normalization_417 (BatchN	(None, 12, 12, 96)	288	conv2d_405[0][0]
activation_402 (Activation)	(None, 12, 12, 384)	0	batch_normalization_414[0][0]
activation_405 (Activation)	(None, 12, 12, 96)	0	batch_normalization_417[0][0]
max_pooling2d_18 (MaxPooling2D)	(None, 12, 12, 288)	0	mixed2[0][0]
mixed3 (Concatenate)	(None, 12, 12, 768)	0	activation_402[0][0] activation_405[0][0] max_pooling2d_18[0][0]
conv2d_410 (Conv2D)	(None, 12, 12, 128)	98304	mixed3[0][0]
batch_normalization_422 (BatchN	(None, 12, 12, 128)	384	conv2d_410[0][0]
activation_410 (Activation)	(None, 12, 12, 128)	0	batch_normalization_422[0][0]
conv2d_411 (Conv2D)	(None, 12, 12, 128)	114688	activation_410[0][0]
batch_normalization_423 (BatchN	(None, 12, 12, 128)	384	conv2d_411[0][0]
activation_411 (Activation)	(None, 12, 12, 128)	0	batch_normalization_423[0][0]
conv2d_407 (Conv2D)	(None, 12, 12, 128)	98304	mixed3[0][0]
conv2d_412 (Conv2D)	(None, 12, 12, 128)	114688	activation_411[0][0]
batch_normalization_419 (BatchN	(None, 12, 12, 128)	384	conv2d_407[0][0]
batch_normalization_424 (BatchN	(None, 12, 12, 128)	384	conv2d_412[0][0]
activation_407 (Activation)	(None, 12, 12, 128)	0	batch_normalization_419[0][0]
activation_412 (Activation)	(None, 12, 12, 128)	0	batch_normalization_424[0][0]
conv2d_408 (Conv2D)	(None, 12, 12, 128)	114688	activation_407[0][0]
conv2d_413 (Conv2D)	(None, 12, 12, 128)	114688	activation_412[0][0]
batch_normalization_420 (BatchN	(None, 12, 12, 128)	384	conv2d_408[0][0]
batch_normalization_425 (BatchN	(None, 12, 12, 128)	384	conv2d_413[0][0]
activation_408 (Activation)	(None, 12, 12, 128)	0	batch_normalization_420[0][0]
activation_413 (Activation)	(None, 12, 12, 128)	0	batch_normalization_425[0][0]
average_pooling2d_39 (AveragePo	(None, 12, 12, 768)	0	mixed3[0][0]
conv2d_406 (Conv2D)	(None, 12, 12, 192)	147456	mixed3[0][0]
conv2d_409 (Conv2D)	(None, 12, 12, 192)	172032	activation_408[0][0]
conv2d_414 (Conv2D)	(None, 12, 12, 192)	172032	activation_413[0][0]
conv2d_415 (Conv2D)	(None, 12, 12, 192)	147456	average_pooling2d_39[0][0]
batch_normalization_418 (BatchN	(None, 12, 12, 192)	576	conv2d_406[0][0]

batch_normalization_421 (BatchN	(None, 12, 12, 192)	576	conv2d_409[0][0]
batch_normalization_426 (BatchN	(None, 12, 12, 192)	576	conv2d_414[0][0]
batch_normalization_427 (BatchN	(None, 12, 12, 192)	576	conv2d_415[0][0]
activation_406 (Activation)	(None, 12, 12, 192)	0	batch_normalization_418[0][0]
activation_409 (Activation)	(None, 12, 12, 192)	0	batch_normalization_421[0][0]
activation_414 (Activation)	(None, 12, 12, 192)	0	batch_normalization_426[0][0]
activation_415 (Activation)	(None, 12, 12, 192)	0	batch_normalization_427[0][0]
mixed4 (Concatenate)	(None, 12, 12, 768)	0	activation_406[0][0] activation_409[0][0] activation_414[0][0] activation_415[0][0]
conv2d_420 (Conv2D)	(None, 12, 12, 160)	122880	mixed4[0][0]
batch_normalization_432 (BatchN	(None, 12, 12, 160)	480	conv2d_420[0][0]
activation_420 (Activation)	(None, 12, 12, 160)	0	batch_normalization_432[0][0]
conv2d_421 (Conv2D)	(None, 12, 12, 160)	179200	activation_420[0][0]
batch_normalization_433 (BatchN	(None, 12, 12, 160)	480	conv2d_421[0][0]
activation_421 (Activation)	(None, 12, 12, 160)	0	batch_normalization_433[0][0]
conv2d_417 (Conv2D)	(None, 12, 12, 160)	122880	mixed4[0][0]
conv2d_422 (Conv2D)	(None, 12, 12, 160)	179200	activation_421[0][0]
batch_normalization_429 (BatchN	(None, 12, 12, 160)	480	conv2d_417[0][0]
batch_normalization_434 (BatchN	(None, 12, 12, 160)	480	conv2d_422[0][0]
activation_417 (Activation)	(None, 12, 12, 160)	0	batch_normalization_429[0][0]
activation_422 (Activation)	(None, 12, 12, 160)	0	batch_normalization_434[0][0]
conv2d_418 (Conv2D)	(None, 12, 12, 160)	179200	activation_417[0][0]
conv2d_423 (Conv2D)	(None, 12, 12, 160)	179200	activation_422[0][0]
batch_normalization_430 (BatchN	(None, 12, 12, 160)	480	conv2d_418[0][0]
batch_normalization_435 (BatchN	(None, 12, 12, 160)	480	conv2d_423[0][0]
activation_418 (Activation)	(None, 12, 12, 160)	0	batch_normalization_430[0][0]
activation_423 (Activation)	(None, 12, 12, 160)	0	batch_normalization_435[0][0]
average_pooling2d_40 (AveragePo	(None, 12, 12, 768)	0	mixed4[0][0]
conv2d_416 (Conv2D)	(None, 12, 12, 192)	147456	mixed4[0][0]
conv2d_419 (Conv2D)	(None, 12, 12, 192)	215040	activation_418[0][0]
conv2d_424 (Conv2D)	(None, 12, 12, 192)	215040	activation_423[0][0]
conv2d_425 (Conv2D)	(None, 12, 12, 192)	147456	average_pooling2d_40[0][0]
batch_normalization_428 (BatchN	(None, 12, 12, 192)	576	conv2d_416[0][0]
batch_normalization_431 (BatchN	(None, 12, 12, 192)	576	conv2d_419[0][0]
batch_normalization_436 (BatchN	(None, 12, 12, 192)	576	conv2d_424[0][0]
batch_normalization_437 (BatchN	(None, 12, 12, 192)	576	conv2d_425[0][0]
activation_416 (Activation)	(None, 12, 12, 192)	0	batch_normalization_428[0][0]
activation_419 (Activation)	(None, 12, 12, 192)	0	batch_normalization_431[0][0]
activation_424 (Activation)	(None, 12, 12, 192)	0	batch_normalization_436[0][0]
activation_425 (Activation)	(None, 12, 12, 192)	0	batch_normalization_437[0][0]
mixed5 (Concatenate)	(None, 12, 12, 768)	0	activation_416[0][0] activation_419[0][0] activation_424[0][0] activation_425[0][0]
conv2d_430 (Conv2D)	(None, 12, 12, 160)	122880	mixed5[0][0]

batch_normalization_442 (BatchN	(None, 12, 12, 160)	480	conv2d_430[0][0]
activation_430 (Activation)	(None, 12, 12, 160)	0	batch_normalization_442[0][0]
conv2d_431 (Conv2D)	(None, 12, 12, 160)	179200	activation_430[0][0]
batch_normalization_443 (BatchN	(None, 12, 12, 160)	480	conv2d_431[0][0]
activation_431 (Activation)	(None, 12, 12, 160)	0	batch_normalization_443[0][0]
conv2d_427 (Conv2D)	(None, 12, 12, 160)	122880	mixed5[0][0]
conv2d_432 (Conv2D)	(None, 12, 12, 160)	179200	activation_431[0][0]
batch_normalization_439 (BatchN	(None, 12, 12, 160)	480	conv2d_427[0][0]
batch_normalization_444 (BatchN	(None, 12, 12, 160)	480	conv2d_432[0][0]
activation_427 (Activation)	(None, 12, 12, 160)	0	batch_normalization_439[0][0]
activation_432 (Activation)	(None, 12, 12, 160)	0	batch_normalization_444[0][0]
conv2d_428 (Conv2D)	(None, 12, 12, 160)	179200	activation_427[0][0]
conv2d_433 (Conv2D)	(None, 12, 12, 160)	179200	activation_432[0][0]
batch_normalization_440 (BatchN	(None, 12, 12, 160)	480	conv2d_428[0][0]
batch_normalization_445 (BatchN	(None, 12, 12, 160)	480	conv2d_433[0][0]
activation_428 (Activation)	(None, 12, 12, 160)	0	batch_normalization_440[0][0]
activation_433 (Activation)	(None, 12, 12, 160)	0	batch_normalization_445[0][0]
average_pooling2d_41 (AveragePo	(None, 12, 12, 768)	0	mixed5[0][0]
conv2d_426 (Conv2D)	(None, 12, 12, 192)	147456	mixed5[0][0]
conv2d_429 (Conv2D)	(None, 12, 12, 192)	215040	activation_428[0][0]
conv2d_434 (Conv2D)	(None, 12, 12, 192)	215040	activation_433[0][0]
conv2d_435 (Conv2D)	(None, 12, 12, 192)	147456	average_pooling2d_41[0][0]
batch_normalization_438 (BatchN	(None, 12, 12, 192)	576	conv2d_426[0][0]
batch_normalization_441 (BatchN	(None, 12, 12, 192)	576	conv2d_429[0][0]
batch_normalization_446 (BatchN	(None, 12, 12, 192)	576	conv2d_434[0][0]
batch_normalization_447 (BatchN	(None, 12, 12, 192)	576	conv2d_435[0][0]
activation_426 (Activation)	(None, 12, 12, 192)	0	batch_normalization_438[0][0]
activation_429 (Activation)	(None, 12, 12, 192)	0	batch_normalization_441[0][0]
activation_434 (Activation)	(None, 12, 12, 192)	0	batch_normalization_446[0][0]
activation_435 (Activation)	(None, 12, 12, 192)	0	batch_normalization_447[0][0]
mixed6 (Concatenate)	(None, 12, 12, 768)	0	activation_426[0][0] activation_429[0][0] activation_434[0][0] activation_435[0][0]
conv2d_440 (Conv2D)	(None, 12, 12, 192)	147456	mixed6[0][0]
batch_normalization_452 (BatchN	(None, 12, 12, 192)	576	conv2d_440[0][0]
activation_440 (Activation)	(None, 12, 12, 192)	0	batch_normalization_452[0][0]
conv2d_441 (Conv2D)	(None, 12, 12, 192)	258048	activation_440[0][0]
batch_normalization_453 (BatchN	(None, 12, 12, 192)	576	conv2d_441[0][0]
activation_441 (Activation)	(None, 12, 12, 192)	0	batch_normalization_453[0][0]
conv2d_437 (Conv2D)	(None, 12, 12, 192)	147456	mixed6[0][0]
conv2d_442 (Conv2D)	(None, 12, 12, 192)	258048	activation_441[0][0]
batch_normalization_449 (BatchN	(None, 12, 12, 192)	576	conv2d_437[0][0]
batch_normalization_454 (BatchN	(None, 12, 12, 192)	576	conv2d_442[0][0]
activation_437 (Activation)	(None, 12, 12, 192)	0	batch_normalization_449[0][0]

activation_442 (Activation)	(None, 12, 12, 192)	0	batch_normalization_454[0][0]
conv2d_438 (Conv2D)	(None, 12, 12, 192)	258048	activation_437[0][0]
conv2d_443 (Conv2D)	(None, 12, 12, 192)	258048	activation_442[0][0]
batch_normalization_450 (BatchN	(None, 12, 12, 192)	576	conv2d_438[0][0]
batch_normalization_455 (BatchN	(None, 12, 12, 192)	576	conv2d_443[0][0]
activation_438 (Activation)	(None, 12, 12, 192)	0	batch_normalization_450[0][0]
activation_443 (Activation)	(None, 12, 12, 192)	0	batch_normalization_455[0][0]
average_pooling2d_42 (AveragePo	(None, 12, 12, 768)	0	mixed6[0][0]
conv2d_436 (Conv2D)	(None, 12, 12, 192)	147456	mixed6[0][0]
conv2d_439 (Conv2D)	(None, 12, 12, 192)	258048	activation_438[0][0]
conv2d_444 (Conv2D)	(None, 12, 12, 192)	258048	activation_443[0][0]
conv2d_445 (Conv2D)	(None, 12, 12, 192)	147456	average_pooling2d_42[0][0]
batch_normalization_448 (BatchN	(None, 12, 12, 192)	576	conv2d_436[0][0]
batch_normalization_451 (BatchN	(None, 12, 12, 192)	576	conv2d_439[0][0]
batch_normalization_456 (BatchN	(None, 12, 12, 192)	576	conv2d_444[0][0]
batch_normalization_457 (BatchN	(None, 12, 12, 192)	576	conv2d_445[0][0]
activation_436 (Activation)	(None, 12, 12, 192)	0	batch_normalization_448[0][0]
activation_439 (Activation)	(None, 12, 12, 192)	0	batch_normalization_451[0][0]
activation_444 (Activation)	(None, 12, 12, 192)	0	batch_normalization_456[0][0]
activation_445 (Activation)	(None, 12, 12, 192)	0	batch_normalization_457[0][0]
mixed7 (Concatenate)	(None, 12, 12, 768)	0	activation_436[0][0] activation_439[0][0] activation_444[0][0] activation_445[0][0]
conv2d_448 (Conv2D)	(None, 12, 12, 192)	147456	mixed7[0][0]
batch_normalization_460 (BatchN	(None, 12, 12, 192)	576	conv2d_448[0][0]
activation_448 (Activation)	(None, 12, 12, 192)	0	batch_normalization_460[0][0]
conv2d_449 (Conv2D)	(None, 12, 12, 192)	258048	activation_448[0][0]
batch_normalization_461 (BatchN	(None, 12, 12, 192)	576	conv2d_449[0][0]
activation_449 (Activation)	(None, 12, 12, 192)	0	batch_normalization_461[0][0]
conv2d_446 (Conv2D)	(None, 12, 12, 192)	147456	mixed7[0][0]
conv2d_450 (Conv2D)	(None, 12, 12, 192)	258048	activation_449[0][0]
batch_normalization_458 (BatchN	(None, 12, 12, 192)	576	conv2d_446[0][0]
batch_normalization_462 (BatchN	(None, 12, 12, 192)	576	conv2d_450[0][0]
activation_446 (Activation)	(None, 12, 12, 192)	0	batch_normalization_458[0][0]
activation_450 (Activation)	(None, 12, 12, 192)	0	batch_normalization_462[0][0]
conv2d_447 (Conv2D)	(None, 5, 5, 320)	552960	activation_446[0][0]
conv2d_451 (Conv2D)	(None, 5, 5, 192)	331776	activation_450[0][0]
batch_normalization_459 (BatchN	(None, 5, 5, 320)	960	conv2d_447[0][0]
batch_normalization_463 (BatchN	(None, 5, 5, 192)	576	conv2d_451[0][0]
activation_447 (Activation)	(None, 5, 5, 320)	0	batch_normalization_459[0][0]
activation_451 (Activation)	(None, 5, 5, 192)	0	batch_normalization_463[0][0]
max_pooling2d_19 (MaxPooling2D)	(None, 5, 5, 768)	0	mixed7[0][0]
mixed8 (Concatenate)	(None, 5, 5, 1280)	0	activation_447[0][0] activation_451[0][0] max_pooling2d_19[0][0]
conv2d_456 (Conv2D)	(None, 5, 5, 448)	573440	mixed8[0][0]

batch_normalization_468 (BatchN	(None, 5, 5, 448)	1344	conv2d_456[0][0]
activation_456 (Activation)	(None, 5, 5, 448)	0	batch_normalization_468[0][0]
conv2d_453 (Conv2D)	(None, 5, 5, 384)	491520	mixed8[0][0]
conv2d_457 (Conv2D)	(None, 5, 5, 384)	1548288	activation_456[0][0]
batch_normalization_465 (BatchN	(None, 5, 5, 384)	1152	conv2d_453[0][0]
batch_normalization_469 (BatchN	(None, 5, 5, 384)	1152	conv2d_457[0][0]
activation_453 (Activation)	(None, 5, 5, 384)	0	batch_normalization_465[0][0]
activation_457 (Activation)	(None, 5, 5, 384)	0	batch_normalization_469[0][0]
conv2d_454 (Conv2D)	(None, 5, 5, 384)	442368	activation_453[0][0]
conv2d_455 (Conv2D)	(None, 5, 5, 384)	442368	activation_453[0][0]
conv2d_458 (Conv2D)	(None, 5, 5, 384)	442368	activation_457[0][0]
conv2d_459 (Conv2D)	(None, 5, 5, 384)	442368	activation_457[0][0]
average_pooling2d_43 (AveragePo	(None, 5, 5, 1280)	0	mixed8[0][0]
conv2d_452 (Conv2D)	(None, 5, 5, 320)	409600	mixed8[0][0]
batch_normalization_466 (BatchN	(None, 5, 5, 384)	1152	conv2d_454[0][0]
batch_normalization_467 (BatchN	(None, 5, 5, 384)	1152	conv2d_455[0][0]
batch_normalization_470 (BatchN	(None, 5, 5, 384)	1152	conv2d_458[0][0]
batch_normalization_471 (BatchN	(None, 5, 5, 384)	1152	conv2d_459[0][0]
conv2d_460 (Conv2D)	(None, 5, 5, 192)	245760	average_pooling2d_43[0][0]
batch_normalization_464 (BatchN	(None, 5, 5, 320)	960	conv2d_452[0][0]
activation_454 (Activation)	(None, 5, 5, 384)	0	batch_normalization_466[0][0]
activation_455 (Activation)	(None, 5, 5, 384)	0	batch_normalization_467[0][0]
activation_458 (Activation)	(None, 5, 5, 384)	0	batch_normalization_470[0][0]
activation_459 (Activation)	(None, 5, 5, 384)	0	batch_normalization_471[0][0]
batch_normalization_472 (BatchN	(None, 5, 5, 192)	576	conv2d_460[0][0]
activation_452 (Activation)	(None, 5, 5, 320)	0	batch_normalization_464[0][0]
mixed9_0 (Concatenate)	(None, 5, 5, 768)	0	activation_454[0][0] activation_455[0][0]
concatenate_8 (Concatenate)	(None, 5, 5, 768)	0	activation_458[0][0] activation_459[0][0]
activation_460 (Activation)	(None, 5, 5, 192)	0	batch_normalization_472[0][0]
mixed9 (Concatenate)	(None, 5, 5, 2048)	0	activation_452[0][0] mixed9_0[0][0] concatenate_8[0][0] activation_460[0][0]
conv2d_465 (Conv2D)	(None, 5, 5, 448)	917504	mixed9[0][0]
batch_normalization_477 (BatchN	(None, 5, 5, 448)	1344	conv2d_465[0][0]
activation_465 (Activation)	(None, 5, 5, 448)	0	batch_normalization_477[0][0]
conv2d_462 (Conv2D)	(None, 5, 5, 384)	786432	mixed9[0][0]
conv2d_466 (Conv2D)	(None, 5, 5, 384)	1548288	activation_465[0][0]
batch_normalization_474 (BatchN	(None, 5, 5, 384)	1152	conv2d_462[0][0]
batch_normalization_478 (BatchN	(None, 5, 5, 384)	1152	conv2d_466[0][0]
activation_462 (Activation)	(None, 5, 5, 384)	0	batch_normalization_474[0][0]
activation_466 (Activation)	(None, 5, 5, 384)	0	batch_normalization_478[0][0]
conv2d_463 (Conv2D)	(None, 5, 5, 384)	442368	activation_462[0][0]
conv2d_464 (Conv2D)	(None, 5, 5, 384)	442368	activation_462[0][0]

conv2d_467 (Conv2D)	(None, 5, 5, 384)	442368	activation_466[0][0]
conv2d_468 (Conv2D)	(None, 5, 5, 384)	442368	activation_466[0][0]
average_pooling2d_44 (AveragePo	(None, 5, 5, 2048)	0	mixed9[0][0]
conv2d_461 (Conv2D)	(None, 5, 5, 320)	655360	mixed9[0][0]
batch_normalization_475 (BatchN	(None, 5, 5, 384)	1152	conv2d_463[0][0]
batch_normalization_476 (BatchN	(None, 5, 5, 384)	1152	conv2d_464[0][0]
batch_normalization_479 (BatchN	(None, 5, 5, 384)	1152	conv2d_467[0][0]
batch_normalization_480 (BatchN	(None, 5, 5, 384)	1152	conv2d_468[0][0]
conv2d_469 (Conv2D)	(None, 5, 5, 192)	393216	average_pooling2d_44[0][0]
batch_normalization_473 (BatchN	(None, 5, 5, 320)	960	conv2d_461[0][0]
activation_463 (Activation)	(None, 5, 5, 384)	0	batch_normalization_475[0][0]
activation_464 (Activation)	(None, 5, 5, 384)	0	batch_normalization_476[0][0]
activation_467 (Activation)	(None, 5, 5, 384)	0	batch_normalization_479[0][0]
activation_468 (Activation)	(None, 5, 5, 384)	0	batch_normalization_480[0][0]
batch_normalization_481 (BatchN	(None, 5, 5, 192)	576	conv2d_469[0][0]
activation_461 (Activation)	(None, 5, 5, 320)	0	batch_normalization_473[0][0]
mixed9_1 (Concatenate)	(None, 5, 5, 768)	0	activation_463[0][0] activation_464[0][0]
concatenate_9 (Concatenate)	(None, 5, 5, 768)	0	activation_467[0][0] activation_468[0][0]
activation_469 (Activation)	(None, 5, 5, 192)	0	batch_normalization_481[0][0]
mixed10 (Concatenate)	(None, 5, 5, 2048)	0	activation_461[0][0] mixed9_1[0][0] concatenate_9[0][0] activation_469[0][0]
reshape_4 (Reshape)	(None, 25, 2048)	0	mixed10[0][0]
lstm_4 (LSTM)	(None, 25, 512)	5244928	reshape_4[0][0]
batch_normalization_482 (BatchN	(None, 25, 512)	2048	lstm_4[0][0]
flatten (Flatten)	(None, 12800)	0	batch_normalization_482[0][0]
dense_12 (Dense)	(None, 4096)	52432896	flatten[0][0]
batch_normalization_483 (BatchN	(None, 4096)	16384	dense_12[0][0]
dense_13 (Dense)	(None, 4096)	16781312	batch_normalization_483[0][0]
batch_normalization_484 (BatchN	(None, 4096)	16384	dense_13[0][0]
dense_14 (Dense)	(None, 4)	16388	batch_normalization_484[0][0]

Total params: 96,313,124
Trainable params: 69,248,004
Non-trainable params: 27,065,120