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/all_equal_300_images/')
```

```
<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]: (1200, 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[300:600]
cataract.shape
```

Out[7]: (300, 2)

In [8]:
```
glaucoma= data[600:900]
glaucoma.shape
```

Out[8]: (300, 2)

In [9]:
```
retina_disease= data[900:1200]
retina_disease.shape
```

Out[9]: (300, 2)

In [10]:
```
random.seed(15)
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]:
```python
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]:
```python
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=30  #specifying the number of images for each class in test phase,calulated as per 10% of total images in each c
lass images 300.


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

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

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

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

    # Now for train and validation set of Normal images first adding 0 to k*n images and then adding all the images fr
om (k+1)*n till last image.

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

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

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

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

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

    # Now for train and validation set of retina disease images first adding 0 to k*n images and then adding all the i
mages from (k+1)*n till last image.

    train_validation_retina= retina_disease[:k*n]
    train_validation_retina= np.append(train_validation_retina,retina_disease[(k+1)*n:],axis=0)
    print('train_validation images for retina disease class from 0 to',k*n,'and',(k+1)*n,'to 300')


    # 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)
    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 30
test images for glaucoma class from 0 30
test images for retina disease class from 0 30
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 30 to 300
train_validation  images for glaucoma class from 0 0 and 30 to 300
train_validation images for retina disease class from 0 to 0 and 30 to 300
model building and compiling for fold 1
Epoch 1/50
27/27 [==============================] - 65s 2s/step - loss: 12.0051 - accuracy: 0.4248 - val_loss: 23.2962 - val_acc
uracy: 0.4398
Epoch 2/50
27/27 [==============================] - 61s 2s/step - loss: 4.9984 - accuracy: 0.5359 - val_loss: 27.8919 - val_accu
racy: 0.2778
Epoch 3/50
27/27 [==============================] - 61s 2s/step - loss: 3.0320 - accuracy: 0.5903 - val_loss: 4.1650 - val_accur
acy: 0.4259
Epoch 4/50
27/27 [==============================] - 58s 2s/step - loss: 1.9502 - accuracy: 0.6343 - val_loss: 3.9679 - val_accur
acy: 0.4398
Epoch 5/50
27/27 [==============================] - 63s 2s/step - loss: 1.4597 - accuracy: 0.7222 - val_loss: 4.7067 - val_accur
acy: 0.4167
Epoch 6/50
27/27 [==============================] - 72s 3s/step - loss: 1.1974 - accuracy: 0.7512 - val_loss: 4.8171 - val_accur
acy: 0.5093
Epoch 7/50
27/27 [==============================] - 69s 3s/step - loss: 0.9097 - accuracy: 0.7905 - val_loss: 3.0465 - val_accur
acy: 0.4769
Epoch 8/50
27/27 [==============================] - 65s 2s/step - loss: 0.5293 - accuracy: 0.8681 - val_loss: 3.2737 - val_accur
acy: 0.4907
Epoch 9/50
27/27 [==============================] - 61s 2s/step - loss: 0.5551 - accuracy: 0.8738 - val_loss: 2.9940 - val_accur
acy: 0.6157
Epoch 10/50
27/27 [==============================] - 61s 2s/step - loss: 0.3649 - accuracy: 0.8958 - val_loss: 4.6517 - val_accur
acy: 0.4444
Epoch 11/50
27/27 [==============================] - 63s 2s/step - loss: 0.4684 - accuracy: 0.9109 - val_loss: 4.9881 - val_accur
acy: 0.5648
Epoch 12/50
27/27 [==============================] - 66s 2s/step - loss: 0.1809 - accuracy: 0.9479 - val_loss: 4.9668 - val_accur
acy: 0.5648
Epoch 13/50
27/27 [==============================] - 64s 2s/step - loss: 0.3526 - accuracy: 0.9248 - val_loss: 5.4506 - val_accur
acy: 0.6157
Epoch 14/50
27/27 [==============================] - 63s 2s/step - loss: 0.2471 - accuracy: 0.9398 - val_loss: 6.2931 - val_accur
acy: 0.4769
Epoch 15/50
27/27 [==============================] - 63s 2s/step - loss: 0.1602 - accuracy: 0.9549 - val_loss: 4.6296 - val_accur
acy: 0.5602
Epoch 16/50
27/27 [==============================] - 64s 2s/step - loss: 0.1526 - accuracy: 0.9468 - val_loss: 4.4881 - val_accur
acy: 0.6528
Epoch 17/50
27/27 [==============================] - 64s 2s/step - loss: 0.1552 - accuracy: 0.9583 - val_loss: 4.8707 - val_accur
acy: 0.6759
Epoch 18/50
27/27 [==============================] - 65s 2s/step - loss: 0.2078 - accuracy: 0.9502 - val_loss: 6.3678 - val_accur
acy: 0.5648
Epoch 19/50
27/27 [==============================] - 65s 2s/step - loss: 0.1522 - accuracy: 0.9676 - val_loss: 2.5349 - val_accur
acy: 0.7083
Epoch 20/50
27/27 [==============================] - 68s 3s/step - loss: 0.1171 - accuracy: 0.9595 - val_loss: 4.0783 - val_accur
acy: 0.7130
Epoch 21/50
27/27 [==============================] - 67s 2s/step - loss: 0.1413 - accuracy: 0.9826 - val_loss: 4.5078 - val_accur
acy: 0.6065
Epoch 22/50
27/27 [==============================] - 67s 2s/step - loss: 0.1296 - accuracy: 0.9711 - val_loss: 4.3435 - val_accur
acy: 0.6574
Epoch 23/50
27/27 [==============================] - 67s 3s/step - loss: 0.1636 - accuracy: 0.9676 - val_loss: 3.3017 - val_accur
acy: 0.7083
Epoch 24/50
27/27 [==============================] - 67s 3s/step - loss: 0.1763 - accuracy: 0.9699 - val_loss: 5.7593 - val_accur
acy: 0.6065
Epoch 25/50
27/27 [==============================] - 67s 3s/step - loss: 0.2042 - accuracy: 0.9630 - val_loss: 3.2316 - val_accur
acy: 0.6852
Epoch 26/50
27/27 [==============================] - 67s 2s/step - loss: 0.0915 - accuracy: 0.9745 - val_loss: 4.6847 - val_accur
acy: 0.6481
```

```
Epoch 27/50
27/27 [==============================] - 68s 3s/step - loss: 0.1183 - accuracy: 0.9711 - val_loss: 3.5744 - val_accur
acy: 0.6528
Epoch 28/50
27/27 [==============================] - 68s 3s/step - loss: 0.0748 - accuracy: 0.9792 - val_loss: 4.3210 - val_accur
acy: 0.6343
Epoch 29/50
27/27 [==============================] - 68s 3s/step - loss: 0.1023 - accuracy: 0.9780 - val_loss: 4.7033 - val_accur
acy: 0.6806
Epoch 30/50
27/27 [==============================] - 67s 2s/step - loss: 0.0935 - accuracy: 0.9769 - val_loss: 4.2346 - val_accur
acy: 0.6435
Epoch 31/50
27/27 [==============================] - 67s 2s/step - loss: 0.1095 - accuracy: 0.9745 - val_loss: 4.2251 - val_accur
acy: 0.6759
Epoch 32/50
27/27 [==============================] - 68s 3s/step - loss: 0.0466 - accuracy: 0.9873 - val_loss: 3.7705 - val_accur
acy: 0.6481
Epoch 33/50
27/27 [==============================] - 68s 3s/step - loss: 0.1216 - accuracy: 0.9792 - val_loss: 5.9386 - val_accur
acy: 0.5880
Epoch 34/50
27/27 [==============================] - 68s 3s/step - loss: 0.0480 - accuracy: 0.9826 - val_loss: 3.3920 - val_accur
acy: 0.6713
Epoch 35/50
27/27 [==============================] - 67s 3s/step - loss: 0.1364 - accuracy: 0.9745 - val_loss: 7.4707 - val_accur
acy: 0.5185
Epoch 36/50
27/27 [==============================] - 69s 3s/step - loss: 0.0114 - accuracy: 0.9965 - val_loss: 4.9823 - val_accur
acy: 0.5741
Epoch 37/50
27/27 [==============================] - 64s 2s/step - loss: 0.1031 - accuracy: 0.9711 - val_loss: 6.7417 - val_accur
acy: 0.5278
Epoch 38/50
27/27 [==============================] - 62s 2s/step - loss: 0.0776 - accuracy: 0.9815 - val_loss: 5.5135 - val_accur
acy: 0.6065
Epoch 39/50
27/27 [==============================] - 64s 2s/step - loss: 0.0857 - accuracy: 0.9803 - val_loss: 4.0091 - val_accur
acy: 0.6759
Epoch 40/50
27/27 [==============================] - 64s 2s/step - loss: 0.0447 - accuracy: 0.9861 - val_loss: 6.5273 - val_accur
acy: 0.6157
Epoch 41/50
27/27 [==============================] - 67s 3s/step - loss: 0.0450 - accuracy: 0.9861 - val_loss: 7.2529 - val_accur
acy: 0.6250
Epoch 42/50
27/27 [==============================] - 68s 3s/step - loss: 0.0501 - accuracy: 0.9896 - val_loss: 3.9146 - val_accur
acy: 0.7176
Epoch 43/50
27/27 [==============================] - 67s 3s/step - loss: 0.0302 - accuracy: 0.9907 - val_loss: 4.5497 - val_accur
acy: 0.6389
Epoch 44/50
27/27 [==============================] - 67s 3s/step - loss: 0.0315 - accuracy: 0.9931 - val_loss: 5.9101 - val_accur
acy: 0.6296
Epoch 45/50
27/27 [==============================] - 66s 2s/step - loss: 0.0327 - accuracy: 0.9896 - val_loss: 6.1181 - val_accur
acy: 0.6250
Epoch 46/50
27/27 [==============================] - 63s 2s/step - loss: 0.0079 - accuracy: 0.9977 - val_loss: 4.5564 - val_accur
acy: 0.6667
Epoch 47/50
27/27 [==============================] - 65s 2s/step - loss: 0.0782 - accuracy: 0.9838 - val_loss: 7.4196 - val_accur
acy: 0.6111
Epoch 48/50
27/27 [==============================] - 71s 3s/step - loss: 0.0186 - accuracy: 0.9954 - val_loss: 6.5377 - val_accur
acy: 0.6111
Epoch 49/50
27/27 [==============================] - 65s 2s/step - loss: 0.0179 - accuracy: 0.9954 - val_loss: 4.9279 - val_accur
acy: 0.7037
Epoch 50/50
27/27 [==============================] - 64s 2s/step - loss: 0.0394 - accuracy: 0.9931 - val_loss: 18.5894 - val_accu
racy: 0.6389
----------------Test accuracy for 1 fold----------------
Confusion Matrix :
 [[18  0  9  3]
 [ 1 28  1  0]
 [ 1  3 24  2]
 [ 2  5  7 16]]
Accuracy    :  0.7166666666666667
Specificity :  0.8853439457869838
Sensitivity :  0.7166666666666666
----------------------------End of 1 Fold-------------------------
------------------------Start of 2 Fold------------------------
test images for normal class from 30 60
test images for cataract class from 30 60
test images for glaucoma class from 30 60
test images for retina disease class from 30 60
train_validation images for normal class from 0 to 30 and 60 to 300
```

```
train_validation images for cataract class from 0 to 30 and 60 to 300
train_validation  images for glaucoma class from 0 30 and 60 to 300
train_validation images for retina disease class from 0 to 30 and 60 to 300
model building and compiling for fold 2
Epoch 1/50
27/27 [==============================] - 71s 2s/step - loss: 10.8944 - accuracy: 0.4549 - val_loss: 20.9369 - val_acc
uracy: 0.3750
Epoch 2/50
27/27 [==============================] - 66s 2s/step - loss: 4.1804 - accuracy: 0.5301 - val_loss: 6.2438 - val_accur
acy: 0.4028
Epoch 3/50
27/27 [==============================] - 66s 2s/step - loss: 2.8937 - accuracy: 0.5556 - val_loss: 7.3541 - val_accur
acy: 0.3843
Epoch 4/50
27/27 [==============================] - 68s 3s/step - loss: 2.7055 - accuracy: 0.6262 - val_loss: 5.5191 - val_accur
acy: 0.3889
Epoch 5/50
27/27 [==============================] - 67s 2s/step - loss: 1.7363 - accuracy: 0.7060 - val_loss: 4.0092 - val_accur
acy: 0.3935
Epoch 6/50
27/27 [==============================] - 67s 2s/step - loss: 1.7728 - accuracy: 0.7419 - val_loss: 2.4735 - val_accur
acy: 0.5000
Epoch 7/50
27/27 [==============================] - 67s 2s/step - loss: 0.9248 - accuracy: 0.8148 - val_loss: 4.5899 - val_accur
acy: 0.3704
Epoch 8/50
27/27 [==============================] - 68s 3s/step - loss: 0.8163 - accuracy: 0.8218 - val_loss: 3.3824 - val_accur
acy: 0.5648
Epoch 9/50
27/27 [==============================] - 68s 3s/step - loss: 1.3710 - accuracy: 0.8484 - val_loss: 2.9249 - val_accur
acy: 0.6481
Epoch 10/50
27/27 [==============================] - 74s 3s/step - loss: 0.4695 - accuracy: 0.9120 - val_loss: 3.0524 - val_accur
acy: 0.5833
Epoch 11/50
27/27 [==============================] - 71s 3s/step - loss: 0.4759 - accuracy: 0.8958 - val_loss: 4.8023 - val_accur
acy: 0.5833
Epoch 12/50
27/27 [==============================] - 69s 3s/step - loss: 0.4575 - accuracy: 0.9132 - val_loss: 3.3315 - val_accur
acy: 0.6435
Epoch 13/50
27/27 [==============================] - 68s 3s/step - loss: 0.2957 - accuracy: 0.9444 - val_loss: 3.1744 - val_accur
acy: 0.6481
Epoch 14/50
27/27 [==============================] - 68s 3s/step - loss: 0.2154 - accuracy: 0.9433 - val_loss: 6.9610 - val_accur
acy: 0.5324
Epoch 15/50
27/27 [==============================] - 63s 2s/step - loss: 0.3159 - accuracy: 0.9294 - val_loss: 4.3049 - val_accur
acy: 0.5787
Epoch 16/50
27/27 [==============================] - 58s 2s/step - loss: 0.1567 - accuracy: 0.9630 - val_loss: 3.6510 - val_accur
acy: 0.6343
Epoch 17/50
27/27 [==============================] - 61s 2s/step - loss: 0.2912 - accuracy: 0.9352 - val_loss: 3.0544 - val_accur
acy: 0.6806
Epoch 18/50
27/27 [==============================] - 63s 2s/step - loss: 0.1643 - accuracy: 0.9560 - val_loss: 7.2186 - val_accur
acy: 0.4861
Epoch 19/50
27/27 [==============================] - 61s 2s/step - loss: 0.2461 - accuracy: 0.9456 - val_loss: 4.7440 - val_accur
acy: 0.6574
Epoch 20/50
27/27 [==============================] - 58s 2s/step - loss: 0.1335 - accuracy: 0.9583 - val_loss: 5.2454 - val_accur
acy: 0.6343
Epoch 21/50
27/27 [==============================] - 56s 2s/step - loss: 0.1732 - accuracy: 0.9618 - val_loss: 3.2853 - val_accur
acy: 0.6806
Epoch 22/50
27/27 [==============================] - 55s 2s/step - loss: 0.1378 - accuracy: 0.9641 - val_loss: 3.6302 - val_accur
acy: 0.6389
Epoch 23/50
27/27 [==============================] - 54s 2s/step - loss: 0.0986 - accuracy: 0.9722 - val_loss: 3.1332 - val_accur
acy: 0.6574
Epoch 24/50
27/27 [==============================] - 56s 2s/step - loss: 0.0792 - accuracy: 0.9792 - val_loss: 3.3248 - val_accur
acy: 0.6389
Epoch 25/50
27/27 [==============================] - 56s 2s/step - loss: 0.0824 - accuracy: 0.9826 - val_loss: 4.7691 - val_accur
acy: 0.6713
Epoch 26/50
27/27 [==============================] - 55s 2s/step - loss: 0.1784 - accuracy: 0.9641 - val_loss: 4.5775 - val_accur
acy: 0.6250
Epoch 27/50
27/27 [==============================] - 56s 2s/step - loss: 0.1417 - accuracy: 0.9745 - val_loss: 3.6571 - val_accur
acy: 0.6713
Epoch 28/50
27/27 [==============================] - 57s 2s/step - loss: 0.0469 - accuracy: 0.9907 - val_loss: 3.5729 - val_accur
acy: 0.6296
```

```
Epoch 29/50
27/27 [==============================] - 56s 2s/step - loss: 0.1363 - accuracy: 0.9722 - val_loss: 6.0048 - val_accur
acy: 0.5741
Epoch 30/50
27/27 [==============================] - 55s 2s/step - loss: 0.0872 - accuracy: 0.9850 - val_loss: 3.6723 - val_accur
acy: 0.6806
Epoch 31/50
27/27 [==============================] - 54s 2s/step - loss: 0.0880 - accuracy: 0.9757 - val_loss: 3.6021 - val_accur
acy: 0.6343
Epoch 32/50
27/27 [==============================] - 53s 2s/step - loss: 0.0834 - accuracy: 0.9803 - val_loss: 3.1608 - val_accur
acy: 0.6898
Epoch 33/50
27/27 [==============================] - 53s 2s/step - loss: 0.1502 - accuracy: 0.9676 - val_loss: 5.3998 - val_accur
acy: 0.6157
Epoch 34/50
27/27 [==============================] - 54s 2s/step - loss: 0.0967 - accuracy: 0.9711 - val_loss: 5.1719 - val_accur
acy: 0.6204
Epoch 35/50
27/27 [==============================] - 54s 2s/step - loss: 0.0827 - accuracy: 0.9780 - val_loss: 3.4785 - val_accur
acy: 0.6898
Epoch 36/50
27/27 [==============================] - 53s 2s/step - loss: 0.0604 - accuracy: 0.9815 - val_loss: 2.9631 - val_accur
acy: 0.7083
Epoch 37/50
27/27 [==============================] - 53s 2s/step - loss: 0.0551 - accuracy: 0.9861 - val_loss: 3.2660 - val_accur
acy: 0.6898
Epoch 38/50
27/27 [==============================] - 53s 2s/step - loss: 0.0684 - accuracy: 0.9873 - val_loss: 4.0356 - val_accur
acy: 0.6620
Epoch 39/50
27/27 [==============================] - 53s 2s/step - loss: 0.0656 - accuracy: 0.9850 - val_loss: 5.0786 - val_accur
acy: 0.6852
Epoch 40/50
27/27 [==============================] - 54s 2s/step - loss: 0.0131 - accuracy: 0.9965 - val_loss: 4.9528 - val_accur
acy: 0.6343
Epoch 41/50
27/27 [==============================] - 54s 2s/step - loss: 0.0743 - accuracy: 0.9861 - val_loss: 6.6387 - val_accur
acy: 0.7176
Epoch 42/50
27/27 [==============================] - 53s 2s/step - loss: 0.0449 - accuracy: 0.9896 - val_loss: 3.8762 - val_accur
acy: 0.6528
Epoch 43/50
27/27 [==============================] - 53s 2s/step - loss: 0.0693 - accuracy: 0.9873 - val_loss: 10.4584 - val_accu
racy: 0.7037
Epoch 44/50
27/27 [==============================] - 55s 2s/step - loss: 0.1309 - accuracy: 0.9722 - val_loss: 22.9637 - val_accu
racy: 0.6944
Epoch 45/50
27/27 [==============================] - 55s 2s/step - loss: 0.0377 - accuracy: 0.9896 - val_loss: 19.9068 - val_accu
racy: 0.7269
Epoch 46/50
27/27 [==============================] - 56s 2s/step - loss: 0.0737 - accuracy: 0.9884 - val_loss: 32.8649 - val_accu
racy: 0.7083
Epoch 47/50
27/27 [==============================] - 57s 2s/step - loss: 0.0321 - accuracy: 0.9942 - val_loss: 22.1987 - val_accu
racy: 0.6944
Epoch 48/50
27/27 [==============================] - 58s 2s/step - loss: 0.0102 - accuracy: 0.9977 - val_loss: 30.4097 - val_accu
racy: 0.6389
Epoch 49/50
27/27 [==============================] - 57s 2s/step - loss: 0.0585 - accuracy: 0.9803 - val_loss: 22.6015 - val_accu
racy: 0.6528
Epoch 50/50
27/27 [==============================] - 55s 2s/step - loss: 0.0721 - accuracy: 0.9850 - val_loss: 24.5505 - val_accu
racy: 0.6759
----------------Test accuracy for 2 fold----------------
Confusion Matrix :
 [[21  1  3  5]
 [ 2 16  6  6]
 [13  0 14  3]
 [ 5  0  6 19]]
Accuracy    :  0.5833333333333334
Specificity :  0.8163277220839988
Sensitivity :  0.5833333333333334
----------------------------End of 2 Fold-------------------------
------------------------Start of 3 Fold------------------------
test images for normal class from 60 90
test images for cataract class from 60 90
test images for glaucoma class from 60 90
test images for retina disease class from 60 90
train_validation images for normal class from 0 to 60 and 90 to 300
train_validation images for cataract class from 0 to 60 and 90 to 300
train_validation  images for glaucoma class from 0 60 and 90 to 300
train_validation images for retina disease class from 0 to 60 and 90 to 300
model building and compiling for fold 3
Epoch 1/50
27/27 [==============================] - 60s 2s/step - loss: 12.7613 - accuracy: 0.4537 - val_loss: 49.8743 - val_acc
```

```
uracy: 0.4028
Epoch 2/50
27/27 [==============================] - 56s 2s/step - loss: 5.4984 - accuracy: 0.5382 - val_loss: 11.7045 - val_accu
racy: 0.3750
Epoch 3/50
27/27 [==============================] - 57s 2s/step - loss: 4.0372 - accuracy: 0.5625 - val_loss: 5.8828 - val_accur
acy: 0.3611
Epoch 4/50
27/27 [==============================] - 57s 2s/step - loss: 2.0472 - accuracy: 0.6562 - val_loss: 3.0645 - val_accur
acy: 0.4213
Epoch 5/50
27/27 [==============================] - 57s 2s/step - loss: 1.3201 - accuracy: 0.7118 - val_loss: 2.9923 - val_accur
acy: 0.5324
Epoch 6/50
27/27 [==============================] - 57s 2s/step - loss: 1.1908 - accuracy: 0.7454 - val_loss: 3.6231 - val_accur
acy: 0.5231
Epoch 7/50
27/27 [==============================] - 58s 2s/step - loss: 0.8101 - accuracy: 0.7951 - val_loss: 3.9090 - val_accur
acy: 0.5231
Epoch 8/50
27/27 [==============================] - 56s 2s/step - loss: 1.3984 - accuracy: 0.8333 - val_loss: 6.3444 - val_accur
acy: 0.4074
Epoch 9/50
27/27 [==============================] - 57s 2s/step - loss: 0.4687 - accuracy: 0.8912 - val_loss: 3.9469 - val_accur
acy: 0.5509
Epoch 10/50
27/27 [==============================] - 57s 2s/step - loss: 0.3977 - accuracy: 0.9062 - val_loss: 7.2188 - val_accur
acy: 0.4861
Epoch 11/50
27/27 [==============================] - 58s 2s/step - loss: 0.3978 - accuracy: 0.9213 - val_loss: 3.7496 - val_accur
acy: 0.6019
Epoch 12/50
27/27 [==============================] - 57s 2s/step - loss: 0.3649 - accuracy: 0.9329 - val_loss: 4.3325 - val_accur
acy: 0.5972
Epoch 13/50
27/27 [==============================] - 58s 2s/step - loss: 0.3100 - accuracy: 0.9340 - val_loss: 5.7684 - val_accur
acy: 0.5278
Epoch 14/50
27/27 [==============================] - 61s 2s/step - loss: 0.2566 - accuracy: 0.9410 - val_loss: 3.6066 - val_accur
acy: 0.5880
Epoch 15/50
27/27 [==============================] - 61s 2s/step - loss: 0.1964 - accuracy: 0.9502 - val_loss: 3.0953 - val_accur
acy: 0.6343
Epoch 16/50
27/27 [==============================] - 59s 2s/step - loss: 0.3185 - accuracy: 0.9468 - val_loss: 4.0929 - val_accur
acy: 0.5972
Epoch 17/50
27/27 [==============================] - 61s 2s/step - loss: 0.3069 - accuracy: 0.9479 - val_loss: 3.0763 - val_accur
acy: 0.6620
Epoch 18/50
27/27 [==============================] - 64s 2s/step - loss: 0.1414 - accuracy: 0.9641 - val_loss: 2.1098 - val_accur
acy: 0.7130
Epoch 19/50
27/27 [==============================] - 64s 2s/step - loss: 0.1674 - accuracy: 0.9606 - val_loss: 3.6543 - val_accur
acy: 0.6435
Epoch 20/50
27/27 [==============================] - 64s 2s/step - loss: 0.2510 - accuracy: 0.9618 - val_loss: 3.7486 - val_accur
acy: 0.6806
Epoch 21/50
27/27 [==============================] - 64s 2s/step - loss: 0.2357 - accuracy: 0.9549 - val_loss: 4.4970 - val_accur
acy: 0.6111
Epoch 22/50
27/27 [==============================] - 65s 2s/step - loss: 0.1209 - accuracy: 0.9630 - val_loss: 3.5828 - val_accur
acy: 0.6296
Epoch 23/50
27/27 [==============================] - 63s 2s/step - loss: 0.1781 - accuracy: 0.9722 - val_loss: 10.1593 - val_accu
racy: 0.4676
Epoch 24/50
27/27 [==============================] - 64s 2s/step - loss: 0.1338 - accuracy: 0.9676 - val_loss: 5.4225 - val_accur
acy: 0.6111
Epoch 25/50
27/27 [==============================] - 61s 2s/step - loss: 0.0595 - accuracy: 0.9861 - val_loss: 8.5245 - val_accur
acy: 0.5278
Epoch 26/50
27/27 [==============================] - 62s 2s/step - loss: 0.2130 - accuracy: 0.9653 - val_loss: 5.4004 - val_accur
acy: 0.6111
Epoch 27/50
27/27 [==============================] - 64s 2s/step - loss: 0.1029 - accuracy: 0.9815 - val_loss: 2.2206 - val_accur
acy: 0.6759
Epoch 28/50
27/27 [==============================] - 63s 2s/step - loss: 0.0735 - accuracy: 0.9826 - val_loss: 2.9416 - val_accur
acy: 0.6481
Epoch 29/50
27/27 [==============================] - 62s 2s/step - loss: 0.1173 - accuracy: 0.9826 - val_loss: 5.3728 - val_accur
acy: 0.5417
Epoch 30/50
27/27 [==============================] - 59s 2s/step - loss: 0.0397 - accuracy: 0.9838 - val_loss: 5.4975 - val_accur
acy: 0.5880
```

```
Epoch 31/50
27/27 [==============================] - 60s 2s/step - loss: 0.0617 - accuracy: 0.9792 - val_loss: 3.4268 - val_accur
acy: 0.6806
Epoch 32/50
27/27 [==============================] - 63s 2s/step - loss: 0.0225 - accuracy: 0.9919 - val_loss: 5.5032 - val_accur
acy: 0.6759
Epoch 33/50
27/27 [==============================] - 62s 2s/step - loss: 0.0597 - accuracy: 0.9850 - val_loss: 7.4036 - val_accur
acy: 0.6204
Epoch 34/50
27/27 [==============================] - 60s 2s/step - loss: 0.1047 - accuracy: 0.9861 - val_loss: 5.1151 - val_accur
acy: 0.6111
Epoch 35/50
27/27 [==============================] - 60s 2s/step - loss: 0.0865 - accuracy: 0.9826 - val_loss: 5.0667 - val_accur
acy: 0.6389
Epoch 36/50
27/27 [==============================] - 59s 2s/step - loss: 0.0430 - accuracy: 0.9896 - val_loss: 5.6837 - val_accur
acy: 0.6620
Epoch 37/50
27/27 [==============================] - 60s 2s/step - loss: 0.0445 - accuracy: 0.9838 - val_loss: 4.6142 - val_accur
acy: 0.6574
Epoch 38/50
27/27 [==============================] - 60s 2s/step - loss: 0.0882 - accuracy: 0.9769 - val_loss: 4.6811 - val_accur
acy: 0.6574
Epoch 39/50
27/27 [==============================] - 61s 2s/step - loss: 0.0760 - accuracy: 0.9861 - val_loss: 2.6446 - val_accur
acy: 0.7593
Epoch 40/50
27/27 [==============================] - 59s 2s/step - loss: 0.0761 - accuracy: 0.9803 - val_loss: 2.4621 - val_accur
acy: 0.7593
Epoch 41/50
27/27 [==============================] - 58s 2s/step - loss: 0.0715 - accuracy: 0.9873 - val_loss: 2.9470 - val_accur
acy: 0.6944
Epoch 42/50
27/27 [==============================] - 58s 2s/step - loss: 0.0418 - accuracy: 0.9850 - val_loss: 4.8909 - val_accur
acy: 0.7361
Epoch 43/50
27/27 [==============================] - 61s 2s/step - loss: 0.0911 - accuracy: 0.9803 - val_loss: 6.0130 - val_accur
acy: 0.6204
Epoch 44/50
27/27 [==============================] - 62s 2s/step - loss: 0.0686 - accuracy: 0.9838 - val_loss: 3.5312 - val_accur
acy: 0.6944
Epoch 45/50
27/27 [==============================] - 59s 2s/step - loss: 0.0586 - accuracy: 0.9907 - val_loss: 3.0837 - val_accur
acy: 0.7222
Epoch 46/50
27/27 [==============================] - 57s 2s/step - loss: 0.0715 - accuracy: 0.9850 - val_loss: 3.7349 - val_accur
acy: 0.6898
Epoch 47/50
27/27 [==============================] - 56s 2s/step - loss: 0.0328 - accuracy: 0.9896 - val_loss: 4.2757 - val_accur
acy: 0.7130
Epoch 48/50
27/27 [==============================] - 58s 2s/step - loss: 0.0253 - accuracy: 0.9942 - val_loss: 8.1234 - val_accur
acy: 0.7269
Epoch 49/50
27/27 [==============================] - 60s 2s/step - loss: 0.0585 - accuracy: 0.9826 - val_loss: 4.1707 - val_accur
acy: 0.7500
Epoch 50/50
27/27 [==============================] - 59s 2s/step - loss: 0.0568 - accuracy: 0.9838 - val_loss: 21.2658 - val_accu
racy: 0.7222
----------------Test accuracy for 3 fold----------------
Confusion Matrix :
 [[26  0  3  1]
 [ 7 23  0  0]
 [ 4  2 21  3]
 [14  2  6  8]]
Accuracy    :  0.65
Specificity :  0.8542775936843734
Sensitivity :  0.6499999999999999
-----------------------------End of 3 Fold-------------------------
------------------------Start of 4 Fold------------------------
test images for normal class from 90 120
test images for cataract class from 90 120
test images for glaucoma class from 90 120
test images for retina disease class from 90 120
train_validation images for normal class from 0 to 90 and 120 to 300
train_validation images for cataract class from 0 to 90 and 120 to 300
train_validation  images for glaucoma class from 0 90 and 120 to 300
train_validation images for retina disease class from 0 to 90 and 120 to 300
model building and compiling for fold 4
Epoch 1/50
27/27 [==============================] - 63s 2s/step - loss: 12.7714 - accuracy: 0.4294 - val_loss: 13.8516 - val_acc
uracy: 0.3889
Epoch 2/50
27/27 [==============================] - 59s 2s/step - loss: 5.0321 - accuracy: 0.5069 - val_loss: 7.5899 - val_accur
acy: 0.5324
Epoch 3/50
27/27 [==============================] - 57s 2s/step - loss: 3.3670 - accuracy: 0.5706 - val_loss: 5.0343 - val_accur
```

```
acy: 0.3750
Epoch 4/50
27/27 [==============================] - 58s 2s/step - loss: 1.9835 - accuracy: 0.6030 - val_loss: 3.9956 - val_accur
acy: 0.4259
Epoch 5/50
27/27 [==============================] - 60s 2s/step - loss: 1.8281 - accuracy: 0.7095 - val_loss: 3.1051 - val_accur
acy: 0.4120
Epoch 6/50
27/27 [==============================] - 62s 2s/step - loss: 1.0048 - accuracy: 0.7975 - val_loss: 5.6232 - val_accur
acy: 0.4167
Epoch 7/50
27/27 [==============================] - 59s 2s/step - loss: 1.1228 - accuracy: 0.7801 - val_loss: 5.5192 - val_accur
acy: 0.4537
Epoch 8/50
27/27 [==============================] - 57s 2s/step - loss: 0.6687 - accuracy: 0.8519 - val_loss: 5.2839 - val_accur
acy: 0.4722
Epoch 9/50
27/27 [==============================] - 58s 2s/step - loss: 0.5905 - accuracy: 0.8738 - val_loss: 7.3839 - val_accur
acy: 0.4074
Epoch 10/50
27/27 [==============================] - 59s 2s/step - loss: 0.5649 - accuracy: 0.8993 - val_loss: 3.8225 - val_accur
acy: 0.5463
Epoch 11/50
27/27 [==============================] - 59s 2s/step - loss: 0.3810 - accuracy: 0.9097 - val_loss: 2.9119 - val_accur
acy: 0.5926
Epoch 12/50
27/27 [==============================] - 60s 2s/step - loss: 0.3983 - accuracy: 0.9178 - val_loss: 3.9343 - val_accur
acy: 0.6065
Epoch 13/50
27/27 [==============================] - 62s 2s/step - loss: 0.3977 - accuracy: 0.9352 - val_loss: 4.7323 - val_accur
acy: 0.6389
Epoch 14/50
27/27 [==============================] - 62s 2s/step - loss: 0.2587 - accuracy: 0.9444 - val_loss: 4.6537 - val_accur
acy: 0.5417
Epoch 15/50
27/27 [==============================] - 61s 2s/step - loss: 0.4075 - accuracy: 0.9201 - val_loss: 5.5771 - val_accur
acy: 0.6157
Epoch 16/50
27/27 [==============================] - 63s 2s/step - loss: 0.1836 - accuracy: 0.9514 - val_loss: 7.9990 - val_accur
acy: 0.5556
Epoch 17/50
27/27 [==============================] - 65s 2s/step - loss: 0.3537 - accuracy: 0.9248 - val_loss: 5.0652 - val_accur
acy: 0.5556
Epoch 18/50
27/27 [==============================] - 64s 2s/step - loss: 0.3385 - accuracy: 0.9421 - val_loss: 7.2024 - val_accur
acy: 0.5556
Epoch 19/50
27/27 [==============================] - 64s 2s/step - loss: 0.2035 - accuracy: 0.9653 - val_loss: 4.8180 - val_accur
acy: 0.6991
Epoch 20/50
27/27 [==============================] - 64s 2s/step - loss: 0.2336 - accuracy: 0.9491 - val_loss: 8.4549 - val_accur
acy: 0.6111
Epoch 21/50
27/27 [==============================] - 64s 2s/step - loss: 0.1263 - accuracy: 0.9641 - val_loss: 4.6292 - val_accur
acy: 0.6667
Epoch 22/50
27/27 [==============================] - 64s 2s/step - loss: 0.1406 - accuracy: 0.9630 - val_loss: 7.3592 - val_accur
acy: 0.6435
Epoch 23/50
27/27 [==============================] - 64s 2s/step - loss: 0.2245 - accuracy: 0.9618 - val_loss: 6.2826 - val_accur
acy: 0.6667
Epoch 24/50
27/27 [==============================] - 64s 2s/step - loss: 0.2335 - accuracy: 0.9630 - val_loss: 7.0821 - val_accur
acy: 0.6898
Epoch 25/50
27/27 [==============================] - 64s 2s/step - loss: 0.0618 - accuracy: 0.9838 - val_loss: 5.7567 - val_accur
acy: 0.7176
Epoch 26/50
27/27 [==============================] - 64s 2s/step - loss: 0.1134 - accuracy: 0.9769 - val_loss: 6.5905 - val_accur
acy: 0.7222
Epoch 27/50
27/27 [==============================] - 64s 2s/step - loss: 0.1862 - accuracy: 0.9688 - val_loss: 9.4272 - val_accur
acy: 0.6991
Epoch 28/50
27/27 [==============================] - 64s 2s/step - loss: 0.1630 - accuracy: 0.9699 - val_loss: 5.4499 - val_accur
acy: 0.5787
Epoch 29/50
27/27 [==============================] - 64s 2s/step - loss: 0.0956 - accuracy: 0.9826 - val_loss: 9.0697 - val_accur
acy: 0.6759
Epoch 30/50
27/27 [==============================] - 65s 2s/step - loss: 0.0723 - accuracy: 0.9826 - val_loss: 4.6815 - val_accur
acy: 0.6991
Epoch 31/50
27/27 [==============================] - 65s 2s/step - loss: 0.1113 - accuracy: 0.9792 - val_loss: 9.1460 - val_accur
acy: 0.5833
Epoch 32/50
27/27 [==============================] - 64s 2s/step - loss: 0.0483 - accuracy: 0.9861 - val_loss: 9.5768 - val_accur
acy: 0.7037
```

```
Epoch 33/50
27/27 [==============================] - 64s 2s/step - loss: 0.1211 - accuracy: 0.9734 - val_loss: 26.3756 - val_accu
racy: 0.6991
Epoch 34/50
27/27 [==============================] - 64s 2s/step - loss: 0.1164 - accuracy: 0.9757 - val_loss: 22.0590 - val_accu
racy: 0.6713
Epoch 35/50
27/27 [==============================] - 64s 2s/step - loss: 0.0794 - accuracy: 0.9792 - val_loss: 16.3455 - val_accu
racy: 0.6620
Epoch 36/50
27/27 [==============================] - 64s 2s/step - loss: 0.1125 - accuracy: 0.9769 - val_loss: 19.4787 - val_accu
racy: 0.6250
Epoch 37/50
27/27 [==============================] - 61s 2s/step - loss: 0.0179 - accuracy: 0.9942 - val_loss: 18.2298 - val_accu
racy: 0.6667
Epoch 38/50
27/27 [==============================] - 59s 2s/step - loss: 0.1033 - accuracy: 0.9838 - val_loss: 30.7828 - val_accu
racy: 0.7037
Epoch 39/50
27/27 [==============================] - 59s 2s/step - loss: 0.0815 - accuracy: 0.9815 - val_loss: 58.2734 - val_accu
racy: 0.6991
Epoch 40/50
27/27 [==============================] - 62s 2s/step - loss: 0.0675 - accuracy: 0.9826 - val_loss: 9.1981 - val_accur
acy: 0.6481
Epoch 41/50
27/27 [==============================] - 61s 2s/step - loss: 0.0428 - accuracy: 0.9861 - val_loss: 15.0244 - val_accu
racy: 0.5741
Epoch 42/50
27/27 [==============================] - 59s 2s/step - loss: 0.0634 - accuracy: 0.9884 - val_loss: 8.1922 - val_accur
acy: 0.6250
Epoch 43/50
27/27 [==============================] - 59s 2s/step - loss: 0.0985 - accuracy: 0.9769 - val_loss: 11.9914 - val_accu
racy: 0.6574
Epoch 44/50
27/27 [==============================] - 61s 2s/step - loss: 0.0049 - accuracy: 0.9977 - val_loss: 10.9101 - val_accu
racy: 0.6898
Epoch 45/50
27/27 [==============================] - 61s 2s/step - loss: 0.0395 - accuracy: 0.9896 - val_loss: 8.7523 - val_accur
acy: 0.5880
Epoch 46/50
27/27 [==============================] - 58s 2s/step - loss: 0.0221 - accuracy: 0.9942 - val_loss: 87.8470 - val_accu
racy: 0.5463
Epoch 47/50
27/27 [==============================] - 59s 2s/step - loss: 0.0713 - accuracy: 0.9873 - val_loss: 85.3573 - val_accu
racy: 0.6157
Epoch 48/50
27/27 [==============================] - 61s 2s/step - loss: 0.0853 - accuracy: 0.9896 - val_loss: 4.2393 - val_accur
acy: 0.6667
Epoch 49/50
27/27 [==============================] - 63s 2s/step - loss: 0.0115 - accuracy: 0.9965 - val_loss: 252.8878 - val_acc
uracy: 0.6667
Epoch 50/50
27/27 [==============================] - 61s 2s/step - loss: 0.0437 - accuracy: 0.9907 - val_loss: 271.4839 - val_acc
uracy: 0.6065
----------------Test accuracy for 4 fold----------------
Confusion Matrix :
 [[16  0  1 13]
 [ 4 18  1  7]
 [ 1  1 15 13]
 [ 8  5  0 17]]
Accuracy    :  0.55
Specificity :  0.8105912022732087
Sensitivity :  0.55
-----------------------------End of 4 Fold-------------------------
------------------------Start of 5 Fold------------------------
test images for normal class from 120 150
test images for cataract class from 120 150
test images for glaucoma class from 120 150
test images for retina disease class from 120 150
train_validation images for normal class from 0 to 120 and 150 to 300
train_validation images for cataract class from 0 to 120 and 150 to 300
train_validation  images for glaucoma class from 0 120 and 150 to 300
train_validation images for retina disease class from 0 to 120 and 150 to 300
model building and compiling for fold 5
Epoch 1/50
27/27 [==============================] - 63s 2s/step - loss: 12.2670 - accuracy: 0.4317 - val_loss: 47.7264 - val_acc
uracy: 0.2407
Epoch 2/50
27/27 [==============================] - 58s 2s/step - loss: 4.4385 - accuracy: 0.5382 - val_loss: 20.1574 - val_accu
racy: 0.3565
Epoch 3/50
27/27 [==============================] - 60s 2s/step - loss: 2.2798 - accuracy: 0.5949 - val_loss: 3.5763 - val_accur
acy: 0.4444
Epoch 4/50
27/27 [==============================] - 62s 2s/step - loss: 1.9193 - accuracy: 0.6435 - val_loss: 3.2520 - val_accur
acy: 0.4537
Epoch 5/50
27/27 [==============================] - 65s 2s/step - loss: 2.1159 - accuracy: 0.6898 - val_loss: 3.5977 - val_accur
```

```
acy: 0.5370
Epoch 6/50
27/27 [==============================] - 64s 2s/step - loss: 1.0072 - accuracy: 0.7685 - val_loss: 4.9058 - val_accur
acy: 0.4583
Epoch 7/50
27/27 [==============================] - 64s 2s/step - loss: 0.8840 - accuracy: 0.8252 - val_loss: 4.3706 - val_accur
acy: 0.5093
Epoch 8/50
27/27 [==============================] - 64s 2s/step - loss: 0.8426 - accuracy: 0.8113 - val_loss: 4.5722 - val_accur
acy: 0.4954
Epoch 9/50
27/27 [==============================] - 65s 2s/step - loss: 0.4337 - accuracy: 0.8866 - val_loss: 4.8491 - val_accur
acy: 0.5185
Epoch 10/50
27/27 [==============================] - 65s 2s/step - loss: 0.2784 - accuracy: 0.9259 - val_loss: 3.6971 - val_accur
acy: 0.5185
Epoch 11/50
27/27 [==============================] - 64s 2s/step - loss: 0.3461 - accuracy: 0.9120 - val_loss: 5.4818 - val_accur
acy: 0.4861
Epoch 12/50
27/27 [==============================] - 64s 2s/step - loss: 0.2475 - accuracy: 0.9294 - val_loss: 2.7369 - val_accur
acy: 0.6065
Epoch 13/50
27/27 [==============================] - 65s 2s/step - loss: 0.2645 - accuracy: 0.9491 - val_loss: 2.7355 - val_accur
acy: 0.6574
Epoch 14/50
27/27 [==============================] - 65s 2s/step - loss: 0.2056 - accuracy: 0.9387 - val_loss: 2.7322 - val_accur
acy: 0.6481
Epoch 15/50
27/27 [==============================] - 64s 2s/step - loss: 0.1105 - accuracy: 0.9664 - val_loss: 7.4019 - val_accur
acy: 0.5000
Epoch 16/50
27/27 [==============================] - 64s 2s/step - loss: 0.2253 - accuracy: 0.9479 - val_loss: 3.1594 - val_accur
acy: 0.6667
Epoch 17/50
27/27 [==============================] - 65s 2s/step - loss: 0.1650 - accuracy: 0.9549 - val_loss: 2.9109 - val_accur
acy: 0.6204
Epoch 18/50
27/27 [==============================] - 65s 2s/step - loss: 0.0762 - accuracy: 0.9815 - val_loss: 3.4394 - val_accur
acy: 0.6435
Epoch 19/50
27/27 [==============================] - 65s 2s/step - loss: 0.1484 - accuracy: 0.9572 - val_loss: 3.0395 - val_accur
acy: 0.6806
Epoch 20/50
27/27 [==============================] - 65s 2s/step - loss: 0.1073 - accuracy: 0.9664 - val_loss: 3.8825 - val_accur
acy: 0.6296
Epoch 21/50
27/27 [==============================] - 65s 2s/step - loss: 0.0958 - accuracy: 0.9745 - val_loss: 4.1132 - val_accur
acy: 0.5972
Epoch 22/50
27/27 [==============================] - 65s 2s/step - loss: 0.0718 - accuracy: 0.9757 - val_loss: 2.6548 - val_accur
acy: 0.6759
Epoch 23/50
27/27 [==============================] - 65s 2s/step - loss: 0.0925 - accuracy: 0.9722 - val_loss: 3.2923 - val_accur
acy: 0.7083
Epoch 24/50
27/27 [==============================] - 65s 2s/step - loss: 0.0727 - accuracy: 0.9826 - val_loss: 2.4204 - val_accur
acy: 0.6667
Epoch 25/50
27/27 [==============================] - 65s 2s/step - loss: 0.0509 - accuracy: 0.9850 - val_loss: 4.4827 - val_accur
acy: 0.5463
Epoch 26/50
27/27 [==============================] - 65s 2s/step - loss: 0.1806 - accuracy: 0.9606 - val_loss: 4.3333 - val_accur
acy: 0.6713
Epoch 27/50
27/27 [==============================] - 64s 2s/step - loss: 0.0695 - accuracy: 0.9815 - val_loss: 4.3806 - val_accur
acy: 0.6250
Epoch 28/50
27/27 [==============================] - 65s 2s/step - loss: 0.0475 - accuracy: 0.9861 - val_loss: 2.9263 - val_accur
acy: 0.6759
Epoch 29/50
27/27 [==============================] - 65s 2s/step - loss: 0.0862 - accuracy: 0.9826 - val_loss: 4.5279 - val_accur
acy: 0.6713
Epoch 30/50
27/27 [==============================] - 65s 2s/step - loss: 0.0617 - accuracy: 0.9884 - val_loss: 5.7775 - val_accur
acy: 0.7037
Epoch 31/50
27/27 [==============================] - 65s 2s/step - loss: 0.0416 - accuracy: 0.9873 - val_loss: 9.8746 - val_accur
acy: 0.6435
Epoch 32/50
27/27 [==============================] - 65s 2s/step - loss: 0.0702 - accuracy: 0.9803 - val_loss: 5.1278 - val_accur
acy: 0.6852
Epoch 33/50
27/27 [==============================] - 65s 2s/step - loss: 0.0619 - accuracy: 0.9896 - val_loss: 6.3339 - val_accur
acy: 0.6389
Epoch 34/50
27/27 [==============================] - 65s 2s/step - loss: 0.0625 - accuracy: 0.9850 - val_loss: 2.7589 - val_accur
acy: 0.6759
```

```
Epoch 35/50
27/27 [==============================] - 65s 2s/step - loss: 0.0250 - accuracy: 0.9919 - val_loss: 11.8915 - val_accu
racy: 0.6806
Epoch 36/50
27/27 [==============================] - 66s 2s/step - loss: 0.0908 - accuracy: 0.9815 - val_loss: 8.0234 - val_accur
acy: 0.5185
Epoch 37/50
27/27 [==============================] - 66s 2s/step - loss: 0.0723 - accuracy: 0.9803 - val_loss: 2.8535 - val_accur
acy: 0.7222
Epoch 38/50
27/27 [==============================] - 65s 2s/step - loss: 0.0648 - accuracy: 0.9850 - val_loss: 3.2334 - val_accur
acy: 0.6620
Epoch 39/50
27/27 [==============================] - 65s 2s/step - loss: 0.0297 - accuracy: 0.9896 - val_loss: 5.2023 - val_accur
acy: 0.6898
Epoch 40/50
27/27 [==============================] - 65s 2s/step - loss: 0.0224 - accuracy: 0.9931 - val_loss: 4.4543 - val_accur
acy: 0.6852
Epoch 41/50
27/27 [==============================] - 65s 2s/step - loss: 0.0505 - accuracy: 0.9907 - val_loss: 14.3069 - val_accu
racy: 0.5833
Epoch 42/50
27/27 [==============================] - 69s 3s/step - loss: 0.0663 - accuracy: 0.9861 - val_loss: 44.0100 - val_accu
racy: 0.6898
Epoch 43/50
27/27 [==============================] - 69s 3s/step - loss: 0.0784 - accuracy: 0.9815 - val_loss: 66.5076 - val_accu
racy: 0.6759
Epoch 44/50
27/27 [==============================] - 63s 2s/step - loss: 0.0527 - accuracy: 0.9896 - val_loss: 16.6842 - val_accu
racy: 0.7130
Epoch 45/50
27/27 [==============================] - 60s 2s/step - loss: 0.0280 - accuracy: 0.9884 - val_loss: 34.5094 - val_accu
racy: 0.6759
Epoch 46/50
27/27 [==============================] - 62s 2s/step - loss: 0.0411 - accuracy: 0.9907 - val_loss: 33.5136 - val_accu
racy: 0.6806
Epoch 47/50
27/27 [==============================] - 64s 2s/step - loss: 0.0625 - accuracy: 0.9884 - val_loss: 18.8493 - val_accu
racy: 0.6667
Epoch 48/50
27/27 [==============================] - 62s 2s/step - loss: 0.0376 - accuracy: 0.9907 - val_loss: 22.4574 - val_accu
racy: 0.6806
Epoch 49/50
27/27 [==============================] - 62s 2s/step - loss: 0.0846 - accuracy: 0.9792 - val_loss: 130.6576 - val_acc
uracy: 0.6667
Epoch 50/50
27/27 [==============================] - 60s 2s/step - loss: 0.0183 - accuracy: 0.9931 - val_loss: 142.8255 - val_acc
uracy: 0.6944
----------------Test accuracy for 5 fold----------------
Confusion Matrix :
 [[18  3  6  3]
 [ 3 21  4  2]
 [ 2  0 27  1]
 [ 2  1  7 20]]
Accuracy    :  0.7166666666666667
Specificity :  0.8854195270785659
Sensitivity :  0.7166666666666666
-----------------------------End of 5 Fold-------------------------
```

Test Evaluation Results

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

Out[17]:
```
[0.7166666666666667, 0.5833333333333334, 0.65, 0.55, 0.7166666666666667]
```

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

Out[18]:
```
0.6433333333333333
```

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

Out[19]:
```
[0.7166666666666666,
 0.5833333333333334,
 0.6499999999999999,
 0.55,
 0.7166666666666666]
```

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

Out[20]:
```
0.6433333333333333
```

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

Out[21]:
```
[0.8853439457869838,
 0.8163277220839988,
 0.8542775936843734,
 0.8105912022732087,
 0.8854195270785659]
```

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

Out[22]:
```
0.850391998181426
```

Training and Validation Evaluation Results

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

Out[23]:
```
array([0.42476851, 0.53587961, 0.59027779, 0.63425928, 0.72222221,
       0.7511574 , 0.79050928, 0.86805558, 0.8738426 , 0.89583331,
       0.91087961, 0.94791669, 0.92476851, 0.93981481, 0.9548611 ,
       0.94675928, 0.95833331, 0.95023149, 0.9675926 , 0.95949072,
       0.9826389 , 0.97106481, 0.9675926 , 0.9699074 , 0.96296299,
       0.97453701, 0.97106481, 0.97916669, 0.97800928, 0.97685188,
       0.97453701, 0.98726851, 0.97916669, 0.9826389 , 0.97453701,
       0.99652779, 0.97106481, 0.98148149, 0.98032409, 0.9861111 ,
       0.9861111 , 0.98958331, 0.99074072, 0.99305558, 0.98958331,
       0.99768519, 0.9837963 , 0.99537039, 0.99537039, 0.99305558,
       0.4548611 , 0.5300926 , 0.55555558, 0.6261574 , 0.70601851,
       0.74189812, 0.81481481, 0.82175928, 0.84837961, 0.91203701,
       0.89583331, 0.91319442, 0.94444442, 0.94328701, 0.92939812,
       0.96296299, 0.93518519, 0.95601851, 0.94560188, 0.95833331,
       0.96180558, 0.96412039, 0.97222221, 0.97916669, 0.9826389 ,
       0.96412039, 0.97453701, 0.99074072, 0.97222221, 0.9849537 ,
       0.97569442, 0.98032409, 0.9675926 , 0.97106481, 0.97800928,
       0.98148149, 0.9861111 , 0.98726851, 0.9849537 , 0.99652779,
       0.9861111 , 0.98958331, 0.98726851, 0.97222221, 0.98958331,
       0.98842591, 0.99421299, 0.99768519, 0.98032409, 0.9849537 ,
       0.4537037 , 0.53819442, 0.5625    , 0.65625   , 0.71180558,
       0.74537039, 0.7951389 , 0.83333331, 0.8912037 , 0.90625   ,
       0.9212963 , 0.93287039, 0.93402779, 0.94097221, 0.95023149,
       0.94675928, 0.94791669, 0.96412039, 0.96064812, 0.96180558,
       0.9548611 , 0.96296299, 0.97222221, 0.9675926 , 0.9861111 ,
       0.96527779, 0.98148149, 0.9826389 , 0.9826389 , 0.9837963 ,
       0.97916669, 0.99189812, 0.9849537 , 0.9861111 , 0.9826389 ,
       0.98958331, 0.9837963 , 0.97685188, 0.9861111 , 0.98032409,
       0.98726851, 0.9849537 , 0.98032409, 0.9837963 , 0.99074072,
       0.9849537 , 0.98958331, 0.99421299, 0.9826389 , 0.9837963 ,
       0.42939815, 0.50694442, 0.57060188, 0.60300928, 0.70949072,
       0.7974537 , 0.7800926 , 0.85185188, 0.8738426 , 0.89930558,
       0.90972221, 0.91782409, 0.93518519, 0.94444442, 0.9201389 ,
       0.9513889 , 0.92476851, 0.94212961, 0.96527779, 0.94907409,
       0.96412039, 0.96296299, 0.96180558, 0.96296299, 0.9837963 ,
       0.97685188, 0.96875   , 0.9699074 , 0.9826389 , 0.9826389 ,
       0.97916669, 0.9861111 , 0.97337961, 0.97569442, 0.97916669,
       0.97685188, 0.99421299, 0.9837963 , 0.98148149, 0.9826389 ,
       0.9861111 , 0.98842591, 0.97685188, 0.99768519, 0.98958331,
       0.99421299, 0.98726851, 0.98958331, 0.99652779, 0.99074072,
       0.43171296, 0.53819442, 0.5949074 , 0.64351851, 0.68981481,
       0.76851851, 0.82523149, 0.8113426 , 0.88657409, 0.92592591,
       0.91203701, 0.92939812, 0.94907409, 0.9386574 , 0.96643519,
       0.94791669, 0.9548611 , 0.98148149, 0.95717591, 0.96643519,
       0.97453701, 0.97569442, 0.97222221, 0.9826389 , 0.9849537 ,
       0.96064812, 0.98148149, 0.9861111 , 0.9826389 , 0.98842591,
       0.98726851, 0.98032409, 0.98958331, 0.9849537 , 0.99189812,
       0.98148149, 0.98032409, 0.9849537 , 0.98958331, 0.99305558,
       0.99074072, 0.9861111 , 0.98148149, 0.98958331, 0.98842591,
       0.99074072, 0.98842591, 0.99074072, 0.97916669, 0.99305558])
```

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

Out[24]:
```
0.9191481482982635
```

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

Out[25]:
```
array([0.43981481, 0.27777779, 0.42592594, 0.43981481, 0.41666666,
       0.50925928, 0.47685185, 0.49074075, 0.61574072, 0.44444445,
       0.56481481, 0.56481481, 0.61574072, 0.47685185, 0.56018519,
       0.65277779, 0.67592591, 0.56481481, 0.70833331, 0.71296299,
       0.60648149, 0.6574074 , 0.70833331, 0.60648149, 0.68518519,
       0.64814812, 0.65277779, 0.63425928, 0.68055558, 0.64351851,
       0.67592591, 0.64814812, 0.58796299, 0.6712963 , 0.51851851,
       0.57407409, 0.52777779, 0.60648149, 0.67592591, 0.61574072,
       0.625     , 0.7175926 , 0.6388889 , 0.62962961, 0.625     ,
       0.66666669, 0.6111111 , 0.6111111 , 0.7037037 , 0.6388889 ,
       0.375     , 0.40277779, 0.38425925, 0.3888889 , 0.39351851,
       0.5       , 0.37037036, 0.56481481, 0.64814812, 0.58333331,
       0.58333331, 0.64351851, 0.64814812, 0.5324074 , 0.5787037 ,
       0.63425928, 0.68055558, 0.4861111 , 0.6574074 , 0.63425928,
       0.68055558, 0.6388889 , 0.6574074 , 0.6388889 , 0.6712963 ,
       0.625     , 0.6712963 , 0.62962961, 0.57407409, 0.68055558,
       0.63425928, 0.68981481, 0.61574072, 0.62037039, 0.68981481,
       0.70833331, 0.68981481, 0.66203701, 0.68518519, 0.63425928,
       0.7175926 , 0.65277779, 0.7037037 , 0.69444442, 0.72685188,
       0.70833331, 0.69444442, 0.6388889 , 0.65277779, 0.67592591,
       0.40277779, 0.375     , 0.3611111 , 0.4212963 , 0.5324074 ,
       0.52314812, 0.52314812, 0.4074074 , 0.55092591, 0.4861111 ,
       0.60185188, 0.59722221, 0.52777779, 0.58796299, 0.63425928,
       0.59722221, 0.66203701, 0.71296299, 0.64351851, 0.68055558,
       0.6111111 , 0.62962961, 0.4675926 , 0.6111111 , 0.52777779,
       0.6111111 , 0.67592591, 0.64814812, 0.54166669, 0.58796299,
       0.68055558, 0.67592591, 0.62037039, 0.6111111 , 0.6388889 ,
       0.66203701, 0.6574074 , 0.6574074 , 0.75925928, 0.75925928,
       0.69444442, 0.7361111 , 0.62037039, 0.69444442, 0.72222221,
       0.68981481, 0.71296299, 0.72685188, 0.75      , 0.72222221,
       0.3888889 , 0.5324074 , 0.375     , 0.42592594, 0.41203704,
       0.41666666, 0.4537037 , 0.47222221, 0.4074074 , 0.5462963 ,
       0.5925926 , 0.60648149, 0.6388889 , 0.54166669, 0.61574072,
       0.55555558, 0.55555558, 0.55555558, 0.69907409, 0.6111111 ,
       0.66666669, 0.64351851, 0.66666669, 0.68981481, 0.7175926 ,
       0.72222221, 0.69907409, 0.5787037 , 0.67592591, 0.69907409,
       0.58333331, 0.7037037 , 0.69907409, 0.6712963 , 0.66203701,
       0.625     , 0.66666669, 0.7037037 , 0.69907409, 0.64814812,
       0.57407409, 0.625     , 0.6574074 , 0.68981481, 0.58796299,
       0.5462963 , 0.61574072, 0.66666669, 0.66666669, 0.60648149,
       0.24074075, 0.35648149, 0.44444445, 0.4537037 , 0.53703701,
       0.45833334, 0.50925928, 0.49537036, 0.51851851, 0.51851851,
       0.4861111 , 0.60648149, 0.6574074 , 0.64814812, 0.5       ,
       0.66666669, 0.62037039, 0.64351851, 0.68055558, 0.62962961,
       0.59722221, 0.67592591, 0.70833331, 0.66666669, 0.5462963 ,
       0.6712963 , 0.625     , 0.67592591, 0.6712963 , 0.7037037 ,
       0.64351851, 0.68518519, 0.6388889 , 0.67592591, 0.68055558,
       0.51851851, 0.72222221, 0.66203701, 0.68981481, 0.68518519,
       0.58333331, 0.68981481, 0.67592591, 0.71296299, 0.67592591,
       0.68055558, 0.66666669, 0.68055558, 0.66666669, 0.69444442])
```

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

Out[26]:
```
0.6060555557012558
```

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

Out[27]:
```
array([1.20050745e+01, 4.99842167e+00, 3.03196573e+00, 1.95022953e+00,
       1.45973837e+00, 1.19738531e+00, 9.09744740e-01, 5.29269457e-01,
       5.55100858e-01, 3.64862055e-01, 4.68422472e-01, 1.80928335e-01,
       3.52628738e-01, 2.47118860e-01, 1.60193622e-01, 1.52577817e-01,
       1.55171961e-01, 2.07757786e-01, 1.52180448e-01, 1.17059521e-01,
       1.41288936e-01, 1.29641652e-01, 1.63623840e-01, 1.76264375e-01,
       2.04210728e-01, 9.15273279e-02, 1.18266404e-01, 7.48287737e-02,
       1.02338403e-01, 9.35271755e-02, 1.09549321e-01, 4.65884507e-02,
       1.21620178e-01, 4.79595810e-02, 1.36431932e-01, 1.14157014e-02,
       1.03143081e-01, 7.76456892e-02, 8.56947079e-02, 4.47275229e-02,
       4.50196788e-02, 5.01108468e-02, 3.01945675e-02, 3.15459333e-02,
       3.26646529e-02, 7.87159614e-03, 7.81944171e-02, 1.86193604e-02,
       1.78637486e-02, 3.94263491e-02, 1.08943624e+01, 4.18041563e+00,
       2.89367938e+00, 2.70554519e+00, 1.73627651e+00, 1.77275443e+00,
       9.24844623e-01, 8.16296875e-01, 1.37100661e+00, 4.69545633e-01,
       4.75907087e-01, 4.57469702e-01, 2.95727223e-01, 2.15383396e-01,
       3.15928727e-01, 1.56739503e-01, 2.91224867e-01, 1.64285660e-01,
       2.46119484e-01, 1.33527920e-01, 1.73213214e-01, 1.37836918e-01,
       9.85804498e-02, 7.91882947e-02, 8.24027359e-02, 1.78383708e-01,
       1.41683057e-01, 4.68709543e-02, 1.36296853e-01, 8.72027054e-02,
       8.80377591e-02, 8.33531842e-02, 1.50191247e-01, 9.67204422e-02,
       8.27496499e-02, 6.04079999e-02, 5.50699383e-02, 6.84329271e-02,
       6.56050220e-02, 1.30570093e-02, 7.42663667e-02, 4.49286364e-02,
       6.92887902e-02, 1.30900726e-01, 3.76681089e-02, 7.36714900e-02,
       3.21283080e-02, 1.02179153e-02, 5.85418679e-02, 7.20790029e-02,
       1.27612848e+01, 5.49844456e+00, 4.03715944e+00, 2.04721689e+00,
       1.32010758e+00, 1.19075286e+00, 8.10090601e-01, 1.39842141e+00,
       4.68676209e-01, 3.97726238e-01, 3.97847593e-01, 3.64851505e-01,
       3.10040891e-01, 2.56595314e-01, 1.96388751e-01, 3.18451136e-01,
       3.06871861e-01, 1.41397119e-01, 1.67445824e-01, 2.51020849e-01,
       2.35678256e-01, 1.20876268e-01, 1.78104684e-01, 1.33825883e-01,
       5.95081560e-02, 2.12998137e-01, 1.02860846e-01, 7.34608173e-02,
       1.17323585e-01, 3.97179350e-02, 6.17191531e-02, 2.24706773e-02,
       5.97232915e-02, 1.04687408e-01, 8.64948332e-02, 4.29531001e-02,
       4.44664955e-02, 8.82124826e-02, 7.59898648e-02, 7.61357173e-02,
       7.14764893e-02, 4.18255292e-02, 9.11304653e-02, 6.86170831e-02,
       5.86182773e-02, 7.14624077e-02, 3.27755772e-02, 2.53029335e-02,
       5.84750213e-02, 5.67858070e-02, 1.27714071e+01, 5.03206825e+00,
       3.36698937e+00, 1.98347950e+00, 1.82806098e+00, 1.00476944e+00,
       1.12278879e+00, 6.68692052e-01, 5.90495884e-01, 5.64880908e-01,
       3.81007016e-01, 3.98294300e-01, 3.97671074e-01, 2.58654743e-01,
       4.07531410e-01, 1.83642611e-01, 3.53693724e-01, 3.38517368e-01,
       2.03490898e-01, 2.33563215e-01, 1.26253039e-01, 1.40562162e-01,
       2.24527285e-01, 2.33527184e-01, 6.17661960e-02, 1.13397948e-01,
       1.86172023e-01, 1.62984595e-01, 9.56119671e-02, 7.23453090e-02,
       1.11257903e-01, 4.82967757e-02, 1.21055491e-01, 1.16426401e-01,
       7.94420689e-02, 1.12513974e-01, 1.79465711e-02, 1.03342384e-01,
       8.14818367e-02, 6.74798638e-02, 4.28090319e-02, 6.33906052e-02,
       9.85353589e-02, 4.86518489e-03, 3.94743681e-02, 2.20602360e-02,
       7.13236704e-02, 8.53476450e-02, 1.15199275e-02, 4.36906889e-02,
       1.22670050e+01, 4.43852234e+00, 2.27977419e+00, 1.91932118e+00,
       2.11594796e+00, 1.00720060e+00, 8.83978724e-01, 8.42586994e-01,
       4.33661014e-01, 2.78396398e-01, 3.46076190e-01, 2.47506723e-01,
       2.64548689e-01, 2.05648795e-01, 1.10450514e-01, 2.25295529e-01,
       1.65021405e-01, 7.62415901e-02, 1.48352116e-01, 1.07337341e-01,
       9.57621560e-02, 7.18374103e-02, 9.25313607e-02, 7.26537853e-02,
       5.08997776e-02, 1.80585489e-01, 6.94516376e-02, 4.74957153e-02,
       8.61655325e-02, 6.16636984e-02, 4.16074768e-02, 7.01758191e-02,
       6.18924946e-02, 6.24638163e-02, 2.49522720e-02, 9.07969996e-02,
       7.23207295e-02, 6.47774115e-02, 2.97177918e-02, 2.23518685e-02,
       5.04515842e-02, 6.63250536e-02, 7.83558711e-02, 5.27344942e-02,
       2.80269664e-02, 4.11267318e-02, 6.25136495e-02, 3.75814922e-02,
       8.46158043e-02, 1.83216259e-02])
```

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

Out[28]:
```
0.6597210306767374
```

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

Out[29]:
```
array([ 23.29616928,  27.89188194,   4.16496325,   3.96786547,
          4.70674562,   4.81713963,   3.04645038,   3.27373409,
          2.99400187,   4.65174818,   4.98810673,   4.96675396,
          5.45057058,   6.29308176,   4.62964725,   4.48814726,
          4.87071133,   6.36779213,   2.53485227,   4.07831621,
          4.50777197,   4.34347486,   3.30170226,   5.75928164,
          3.2316494 ,   4.68473434,   3.57436562,   4.32099915,
          4.70328617,   4.23464584,   4.22507048,   3.7704854 ,
          5.93859339,   3.39202285,   7.47074366,   4.98230267,
          6.74173212,   5.51354694,   4.00906706,   6.52734089,
          7.25287247,   3.91458225,   4.54965258,   5.91009855,
          6.11809301,   4.55641603,   7.41960764,   6.53773499,
          4.92791176,  18.58941269,  20.93690109,   6.24384737,
          7.35406303,   5.51905394,   4.00919771,   2.47350907,
          4.58988237,   3.38239455,   2.92489743,   3.05239701,
          4.80225563,   3.3315289 ,   3.17443037,   6.96102715,
          4.30493021,   3.65098691,   3.0544219 ,   7.21863031,
          4.74396992,   5.24535036,   3.28530288,   3.63020587,
          3.13315845,   3.32476139,   4.76905489,   4.57752895,
          3.65713048,   3.57292724,   6.00475454,   3.67228508,
          3.60207391,   3.16084433,   5.39980221,   5.17192602,
          3.47846866,   2.96308589,   3.26603723,   4.03562975,
          5.07859612,   4.95282078,   6.63868141,   3.87624311,
         10.45837021,  22.96370125,  19.90676117,  32.86488724,
         22.19865417,  30.40971184,  22.60146713,  24.55046082,
         49.8742981 ,  11.70446014,   5.88280869,   3.0644536 ,
          2.99234056,   3.6231339 ,   3.90901899,   6.34442472,
          3.94692779,   7.21878147,   3.74964571,   4.3325491 ,
          5.76836634,   3.60655785,   3.09528327,   4.09285021,
          3.07626772,   2.10979629,   3.65431547,   3.74864221,
          4.49695778,   3.58281565,  10.1593256 ,   5.42247868,
          8.52452564,   5.40042734,   2.22061896,   2.94159818,
          5.37275743,   5.49750471,   3.42680359,   5.50315332,
          7.40359497,   5.11507463,   5.06669331,   5.68367863,
          4.61419773,   4.68109083,   2.64456224,   2.46208382,
          2.94703031,   4.89092922,   6.01302004,   3.53122234,
          3.08367109,   3.73489499,   4.27570105,   8.1234169 ,
          4.17074823,  21.26581764,  13.85161209,   7.58985949,
          5.03430128,   3.99562716,   3.10505939,   5.62321091,
          5.51918983,   5.28388309,   7.3838768 ,   3.82252979,
          2.91187477,   3.93432736,   4.732265  ,   4.65366602,
          5.57710028,   7.99897242,   5.06519699,   7.2024107 ,
          4.81804323,   8.45487499,   4.62921238,   7.35918951,
          6.28257227,   7.0820961 ,   5.75667238,   6.59050512,
          9.42719078,   5.44993925,   9.06974888,   4.68148136,
          9.14601231,   9.57682991,  26.37563133,  22.05903816,
         16.34545898,  19.47872734,  18.2297821 ,  30.7828331 ,
         58.27344513,   9.19807911,  15.02436161,   8.19217777,
         11.99136734,  10.9100647 ,   8.75228786,  87.84699249,
         85.35726929,   4.23925972, 252.88783264, 271.48394775,
         47.72638321,  20.15737724,   3.57625437,   3.25197864,
          3.59768414,   4.90584946,   4.37056017,   4.57223606,
          4.84905386,   3.6971004 ,   5.48178911,   2.73685551,
          2.7354517 ,   2.73218036,   7.40186977,   3.15936637,
          2.91088438,   3.43936014,   3.03950214,   3.88246489,
          4.113204  ,   2.65477586,   3.29230475,   2.42039132,
          4.48271275,   4.33325291,   4.38058043,   2.92626405,
          4.52786493,   5.77745104,   9.87459946,   5.12780857,
          6.3338871 ,   2.75890183,  11.89151859,   8.0233736 ,
          2.85346818,   3.23340058,   5.20227623,   4.45434904,
         14.30693722,  44.00995255,  66.50756073,  16.68421555,
         34.50941086,  33.51356506,  18.84931374,  22.45737457,
        130.65757751, 142.82551575])
```

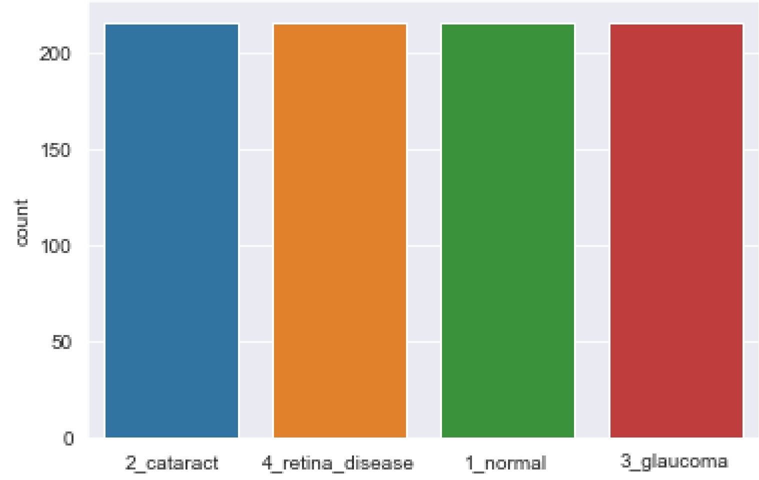
In [30]:
```
mean_val_loss=np.mean(val_loss)
mean_val_loss
```

Out[30]:
```
11.595856408119202
```

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

In [31]:
```python
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)
```

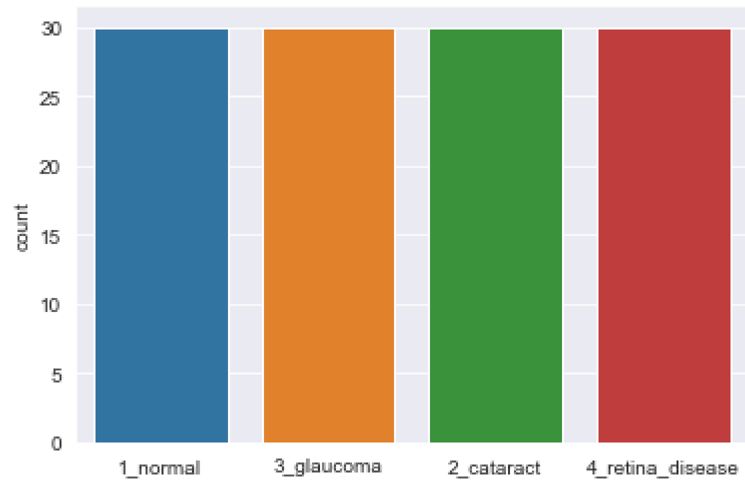
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x2018da24c10>



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

In [32]:
```python
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)
```

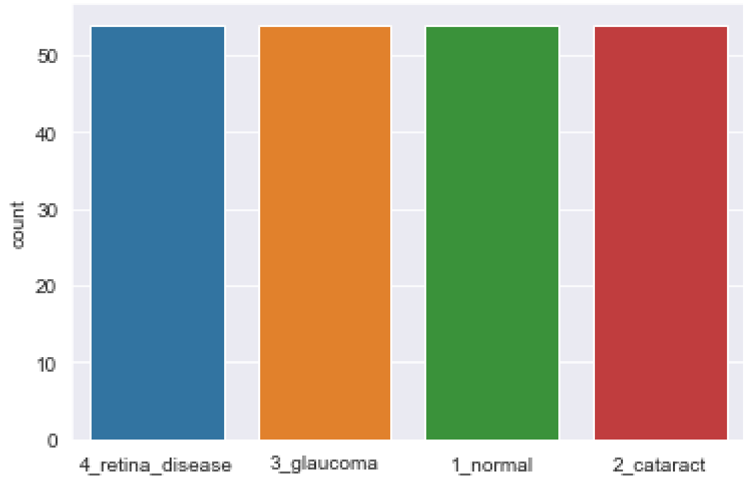
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x201c0b19460>



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

In [33]:
```python
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 0x201a876f730>



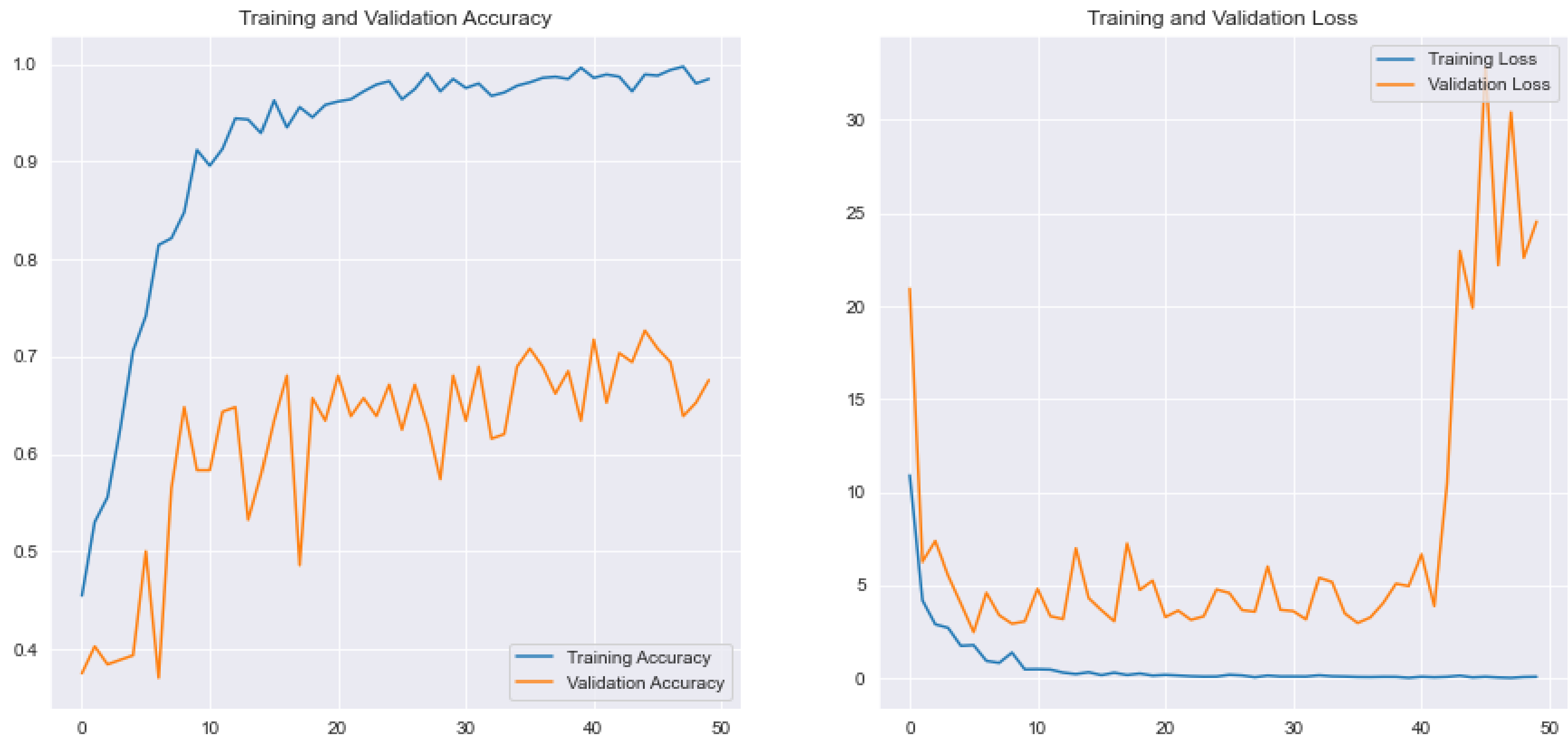
Training,Validation Accuracy and Loss Plot for 50 Epochs

In [34]:
```python
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)
```

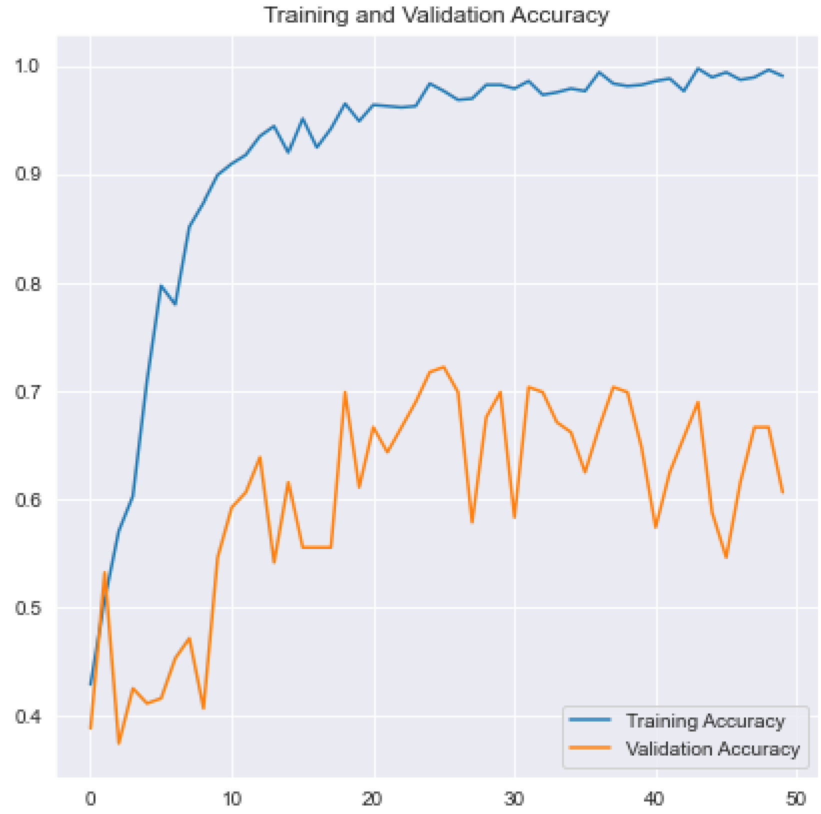
Plot for 1 cross validation accuracy and loss for Training and Validation phase

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

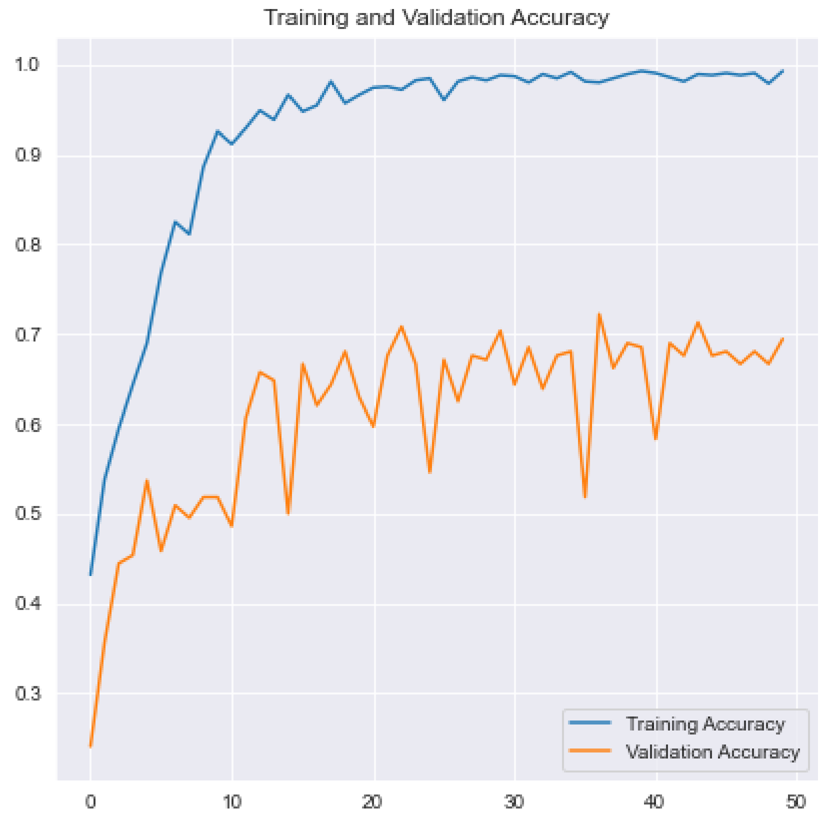


Plot for 3 cross validation accuracy and loss for Training and Validation phase

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

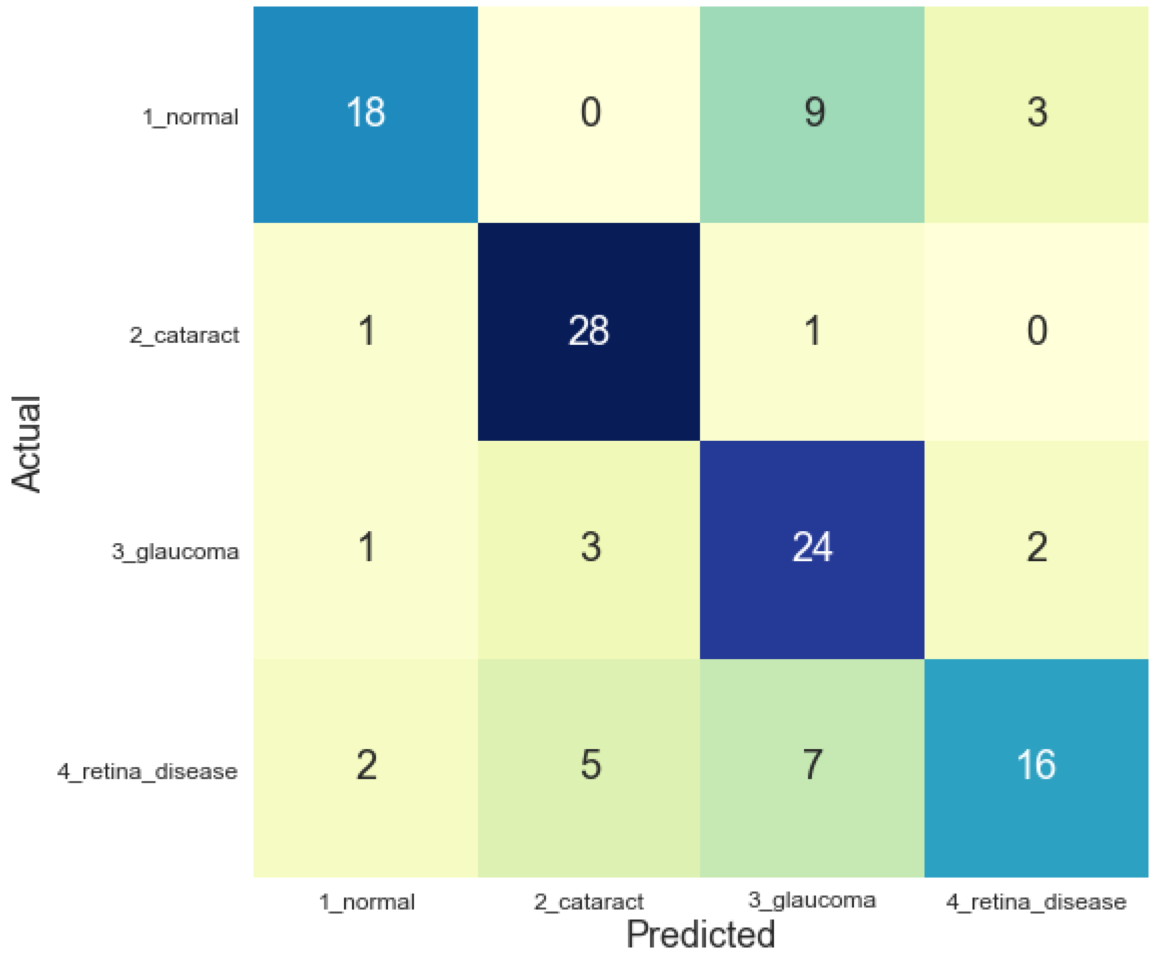
In [36]:
```
CM= np.array(CM)
CM.resize(5,4,4)
```

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()
```

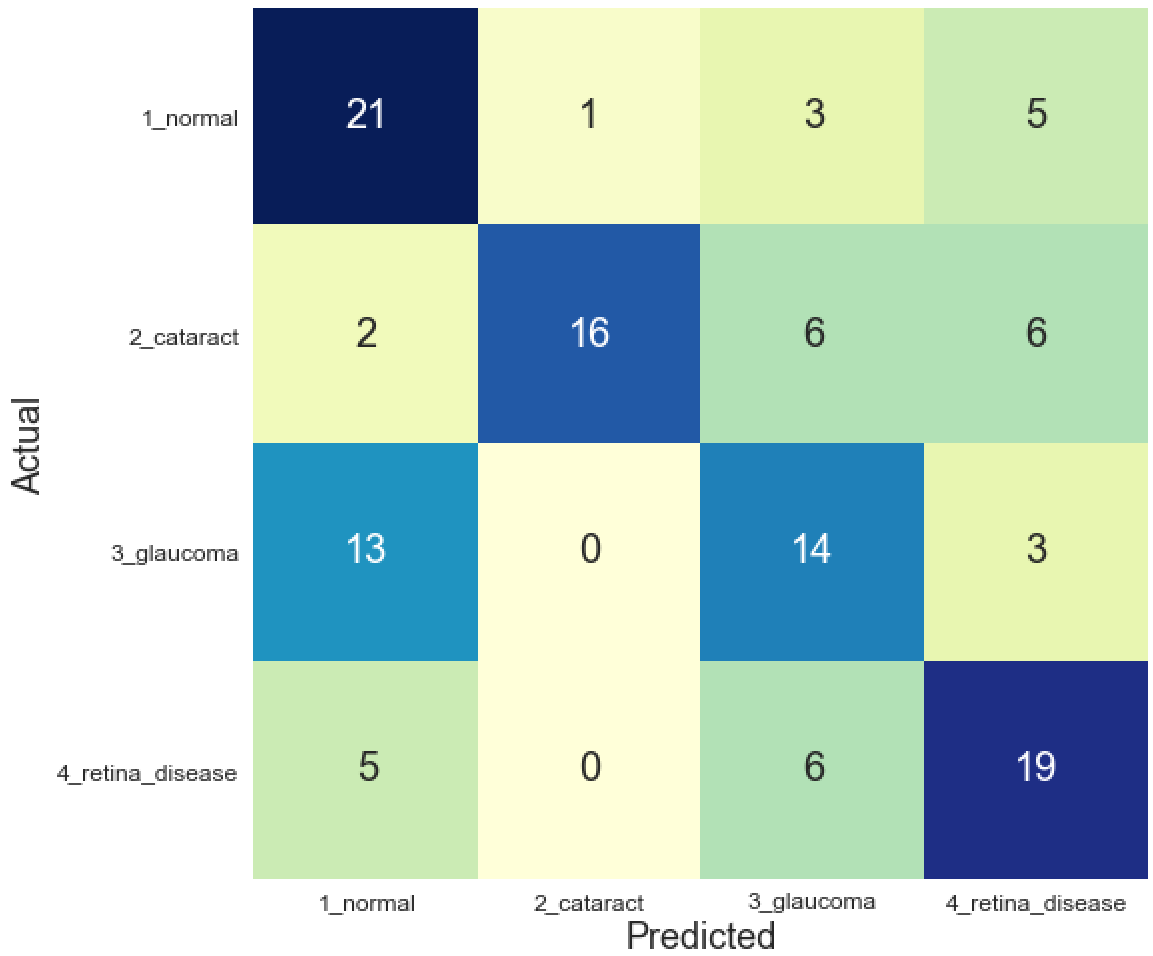
In [38]:

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

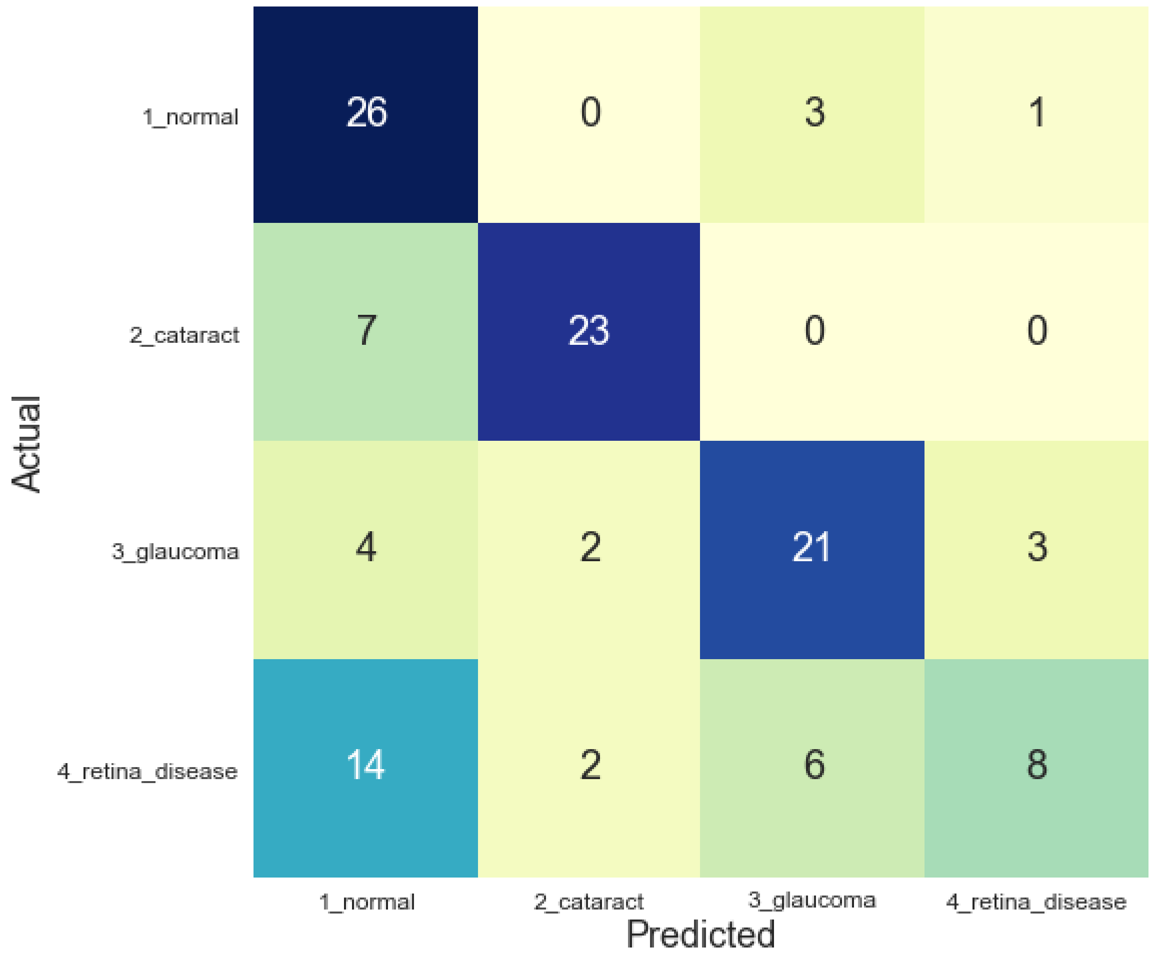
Confusion Matrix for 1 Cross Validation Test phase



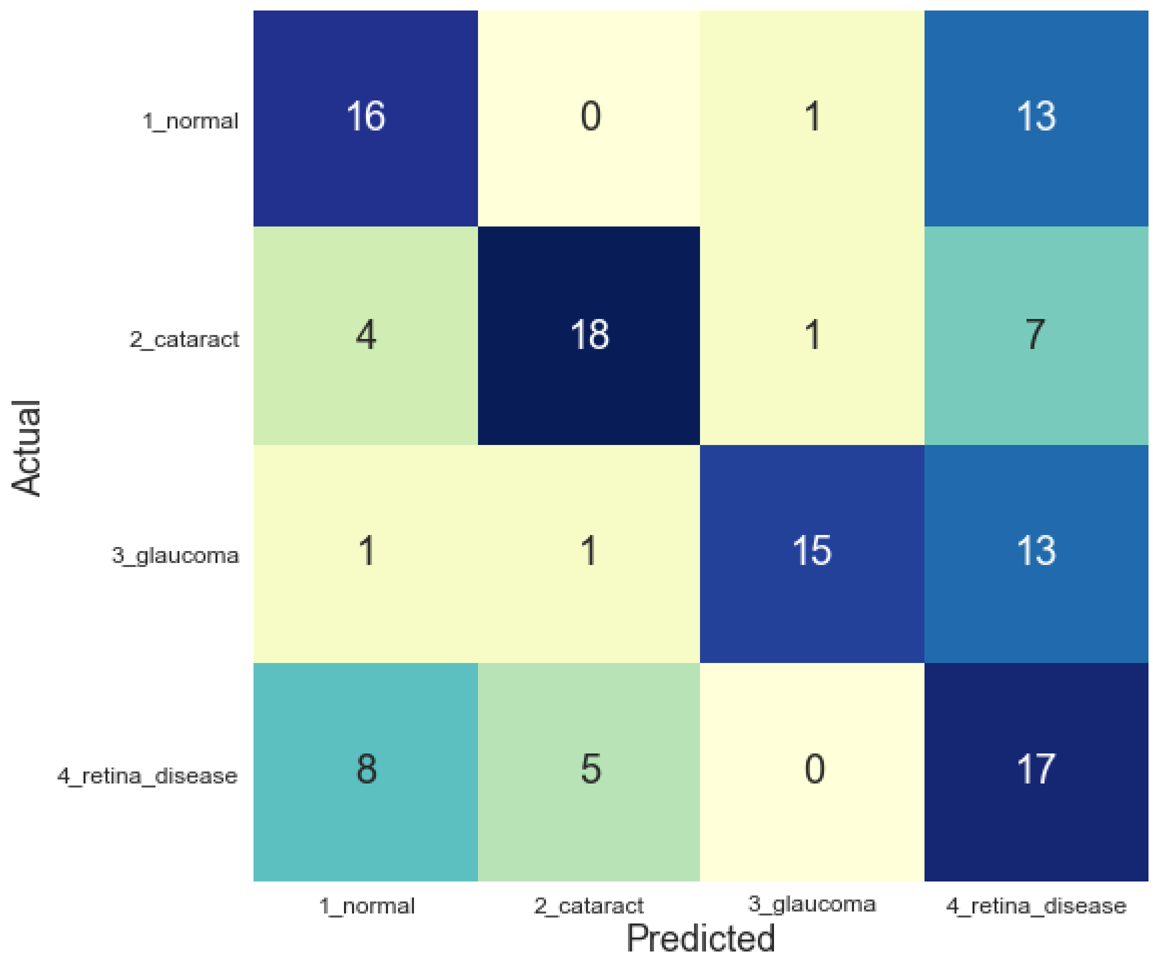
Confusion Matrix for 2 Cross Validation Test phase



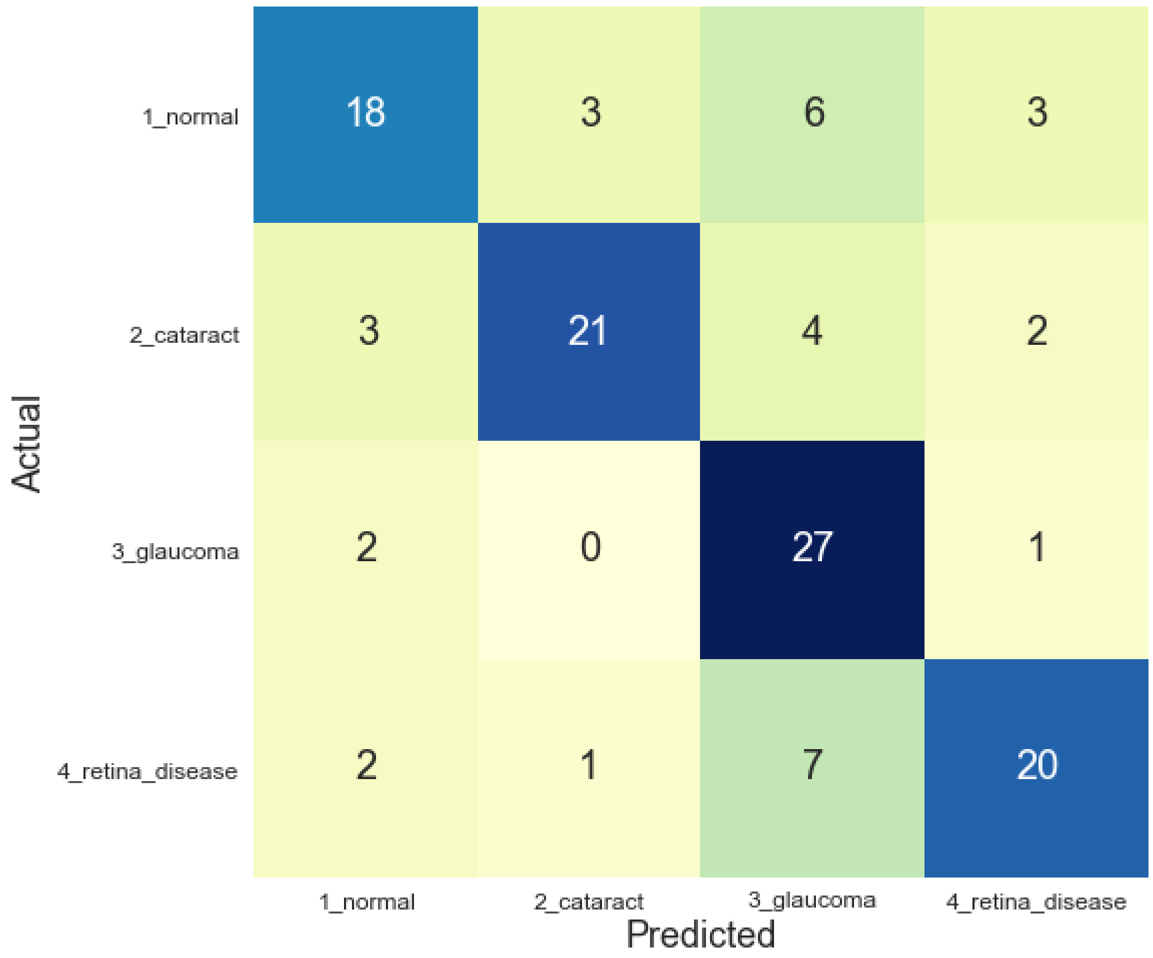
Confusion Matrix for 3 Cross Validation Test phase



Confusion Matrix for 4 Cross Validation Test phase



Confusion Matrix for 5 Cross Validation Test phase

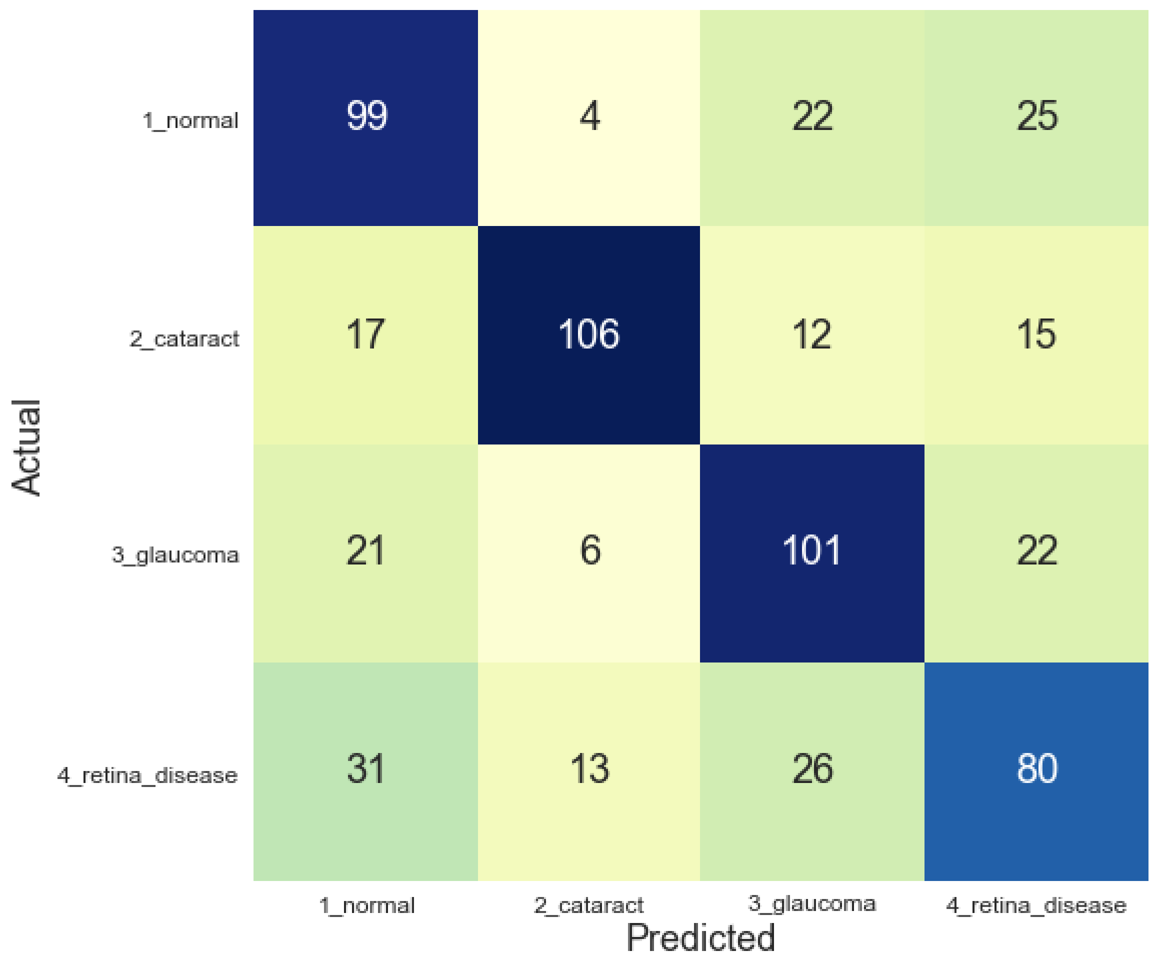


Visualizing Summarized Confusion Matrix of all 5 folds

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

Out[39]:
```
array([[ 99,   4,  22,  25],
       [ 17, 106,  12,  15],
       [ 21,   6, 101,  22],
       [ 31,  13,  26,  80]], dtype=int64)
```

In [40]:
```
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 [41]:

```
    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    :  0.6433333333333333
Specificity :  0.8469702200434992
Sensitivity :  0.6433333333333333
```

Model Summary

In [42]:

```
model_build_compile(k)
```

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

Out[42]:

```
<tensorflow.python.keras.engine.functional.Functional at 0x2024ba6ffa0>
```

In [43]:

```
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
__________________________________________________________________________________________________
```