Data Collection

In [1]:

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

file_id = '1WkCyl3kFta2GwV2ZJes0ZNNFNbfBl1LE'

# YD_dataset 1WkCyl3kFta2GwV2ZJes0ZNNFNbfBl1LE
# Yawn_dataset 1LE3BHEpRuIDe-yedGriGg1Ix6_ZliEsp
downloaded = drive.CreateFile({'id': file_id})
downloaded.GetContentFile('yawn_dataset.zip')
```

In [2]:

```
!unzip yawn_dataset.zip
```

```
Archive:  yawn_dataset.zip
   creating: YD_dataset/test/
   creating: YD_dataset/test/no_yawn/
  inflating: YD_dataset/test/no_yawn/1004.jpg
  inflating: YD_dataset/test/no_yawn/1007.jpg
  inflating: YD_dataset/test/no_yawn/1010.jpg
  inflating: YD_dataset/test/no_yawn/1033.jpg
  inflating: YD_dataset/test/no_yawn/1044.jpg
  inflating: YD_dataset/test/no_yawn/1050.jpg
  inflating: YD_dataset/test/no_yawn/1063.jpg
  inflating: YD_dataset/test/no_yawn/1067.jpg
  inflating: YD_dataset/test/no_yawn/1096.jpg
  inflating: YD_dataset/test/no_yawn/1114.jpg
  inflating: YD_dataset/test/no_yawn/1118.jpg
  inflating: YD_dataset/test/no_yawn/1129.jpg
  inflating: YD_dataset/test/no_yawn/113.jpg
  inflating: YD_dataset/test/no_yawn/1134.jpg
  inflating: YD_dataset/test/no_yawn/115.jpg
  inflating: YD_dataset/test/no_yawn/1213.jpg
```

Data Augmentation

In [3]:

```
import numpy as  np
import pandas as pd

train = pd.read_csv('csv_dataset.csv')
train
```

Out[3]:

	image_names	yawn_or_not
0	YD_dataset/train/yawn/1.jpg	1
1	YD_dataset/train/yawn/10.jpg	1
2	YD_dataset/train/yawn/101.jpg	1
3	YD_dataset/train/yawn/103.jpg	1
4	YD_dataset/train/yawn/104.jpg	1
...	...	...
1443	YD_dataset/train/no_yawn/992.jpg	0
1444	YD_dataset/train/no_yawn/993.jpg	0
1445	YD_dataset/train/no_yawn/994.jpg	0
1446	YD_dataset/train/no_yawn/997.jpg	0
1447	YD_dataset/train/no_yawn/998.jpg	0

1448 rows × 2 columns

In [4]:

```
from skimage.io import imread
from skimage.transform import resize
import matplotlib.pyplot as plt
%matplotlib inline



train_img = []
for img_name in train['image_names']:
    # defining the image path
    image_path = '/content/' + img_name
    # reading the image
    img = imread(image_path)
    # normalizing the pixel values
    img = img/255
    # resizing the image to (224,224,3)
    img = resize(img, output_shape=(224,224,3), mode='constant', anti_aliasing=True)
    # converting the type of pixel to float 32
    img = img.astype('float32')
    # appending the image into the list
    train_img.append(img)

images = np.array(train_img)
images.shape
```

Out[4]:

```
(1448, 224, 224, 3)
```

In [5]:

```
labels = train['yawn_or_not'].values
labels.shape
```

Out[5]:

```
(1448,)
```

In [6]:

```
labels
```

Out[6]:

```
array([1, 1, 1, ..., 0, 0, 0])
```

In [7]:

```
from sklearn.model_selection import train_test_split

Xtrain,Xtest,Ytrain,Ytest = train_test_split(images,labels, test_size = 0.148, random_state

(Xtrain.shape, Ytrain.shape), (Xtest.shape, Ytest.shape)
```

Out[7]:

```
(((1233, 224, 224, 3), (1233,)), ((215, 224, 224, 3), (215,)))
```

In [8]:

```
Ytest_bin = Ytest
Ytest_bin
```

Out[8]:

```
array([0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1])
```

In [9]:

```
from keras.utils import to_categorical

Ytrain = to_categorical(Ytrain)
Ytrain
```

Out[9]:

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

In [10]:

```
Ytest = to_categorical(Ytest)
Ytest
```

Out[10]:

```
array([[1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
```

In [11]:

```
print(Ytest.shape,Ytest_bin.shape)
```

```
(215, 2) (215,)
```

Building model architecture

In [14]:

```
import tensorflow as tf
from keras.models import Sequential
from keras import layers
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation,GlobalMa
from keras import applications
from keras.applications import VGG16
from keras.models import Model
from keras import optimizers



base_model = VGG16(input_shape = (224, 224, 3),
include_top = False,
weights = 'imagenet')


for layer in base_model.layers:
    layer.trainable = False

x = layers.Flatten()(base_model.output)
x = layers.Dense(512, activation='relu')(x)

x = layers.Dropout(0.5)(x)
x = layers.Dense(2, activation='softmax')(x)

model = tf.keras.models.Model(base_model.input, x)

model.compile(optimizer = tf.keras.optimizers.RMSprop(lr=0.0001), loss = 'binary_crossentro
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
```

```
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 512)               12845568
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1026
=================================================================
Total params: 27,561,282
Trainable params: 12,846,594
Non-trainable params: 14,714,688
_________________________________________________________________
```

Model Training

In [15]:

```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint("model_weights.h5", monitor='val_acc', verbose=1, save_best_on
callbacks_list = [checkpoint]
```

In [16]:

```
training = model.fit(Xtrain,Ytrain,callbacks=callbacks_list,validation_data=(Xtest,Ytest),e
```

```
Epoch 1/50
39/39 [==============================] - 54s 392ms/step - loss: 1.0809 - a
cc: 0.5795 - val_loss: 0.5304 - val_acc: 0.6372

Epoch 00001: val_acc improved from -inf to 0.63721, saving model to model_
weights.h5
Epoch 2/50
39/39 [==============================] - 7s 168ms/step - loss: 0.5094 - ac
c: 0.7302 - val_loss: 0.5431 - val_acc: 0.6744

Epoch 00002: val_acc improved from 0.63721 to 0.67442, saving model to mod
el_weights.h5
Epoch 3/50
39/39 [==============================] - 7s 169ms/step - loss: 0.3921 - ac
c: 0.8209 - val_loss: 0.3999 - val_acc: 0.7860

Epoch 00003: val_acc improved from 0.67442 to 0.78605, saving model to mod
el_weights.h5
Epoch 4/50
```

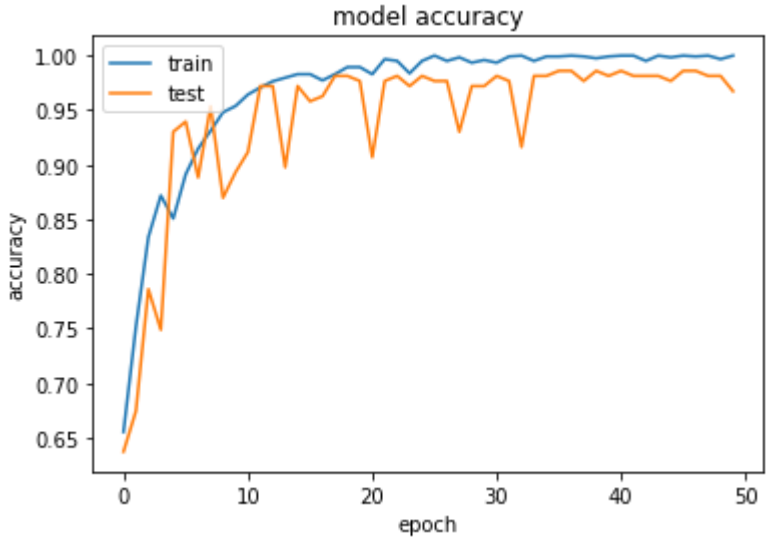
Model Evaluation

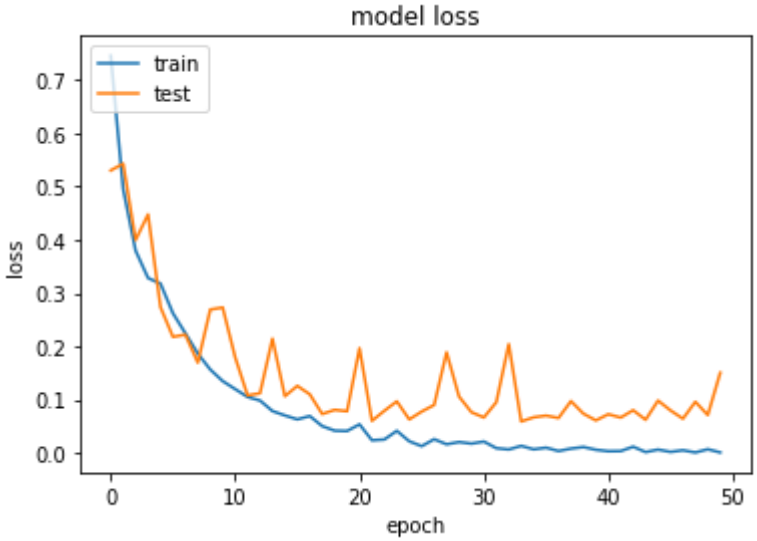
In [18]:

```
import matplotlib.pyplot as plt

# summarize training for accuracy
plt.plot(training.history['acc'])
plt.plot(training.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()

# summarize traning for loss
plt.plot(training.history['loss'])
plt.plot(training.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```





In [17]:

```
result = model.evaluate(Xtest,Ytest,verbose=1)

print("test loss : ",result[0])
print("test acc. :",result[1])
```

```
7/7 [==============================] - 1s 130ms/step - loss: 0.1505 - acc:
0.9674
test loss :  0.1505407989025116
test acc. : 0.9674418568611145
```

In [19]:

```
predictions= model.predict(Xtest,verbose=1)
predicted_classes = np.argmax(predictions,axis=1)
```

```
7/7 [==============================] - 1s 155ms/step
```

In [20]:

```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score

print(accuracy_score(Ytest_bin,predicted_classes))
```

```
0.9674418604651163
```

In [21]:

```
print(confusion_matrix(Ytest_bin,predicted_classes))
```

```
[[102   0]
 [  7 106]]
```

In [22]:

```
print(classification_report(Ytest_bin,predicted_classes,target_names=["yawn","no_yawn"]))
```

```
              precision    recall  f1-score   support

        yawn       0.94      1.00      0.97       102
     no_yawn       1.00      0.94      0.97       113

    accuracy                           0.97       215
   macro avg       0.97      0.97      0.97       215
weighted avg       0.97      0.97      0.97       215
```

In [23]:

```
training.history.keys()
```

Out[23]:

```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

In [24]:

```
from sklearn.metrics import roc_curve,roc_auc_score

roc_auc_score(Ytest_bin,predicted_classes)
```

Out[24]:

```
0.9690265486725664
```

In [25]:

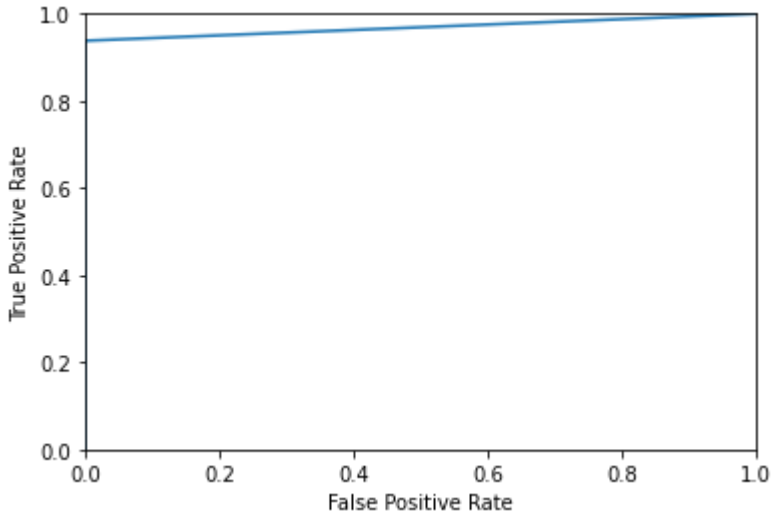
```
fpr , tpr , thresholds = roc_curve(Ytest_bin,predicted_classes)
```

In [26]:

```
import matplotlib.pyplot as plt

def plot_roc_curve(fpr,tpr):
  plt.plot(fpr,tpr)
  plt.axis([0,1,0,1])
  plt.xlabel('False Positive Rate')
  plt.ylabel('True Positive Rate')
  plt.show()

plot_roc_curve(fpr,tpr)
```



Saving model

In [ ]:

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

In [ ]:

```
model.save("Yawn_classifier.model")
```

In [ ]:

```
!zip -r /content/model.zip /content/Yawn_classifier.model
```

In [ ]:

```
"""
!wget http://skulddata.cs.umass.edu/traces/mmsys/2014/user06.tar
!tar -xvf /content/user06.tar
!pip install patool
!pip install unrar
import patoolib
patoolib.extract_archive("/content/user06/YawDD dataset.rar", outdir="/content/")
"""
```