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 [1]:

```
import numpy as  np
import pandas as pd

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

Out[1]:

	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 [2]:

```
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[2]:

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

In [3]:

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

Out[3]:

```
(1448,)
```

In [4]:

```
labels
```

Out[4]:

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

In [17]:

```

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[17]:

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

In [18]:

```
Ytest_bin = Ytest
Ytest_bin
```

Out[18]:

```
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 [19]:

```
from keras.utils import to_categorical

Ytrain = to_categorical(Ytrain)
Ytrain
```

Out[19]:

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

In [20]:

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

Out[20]:

```
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 [21]:

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

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

Building model architecture

In [14]:

```
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

input_shape = (224,224, 3)


pre_trained_model = VGG16(input_shape=input_shape, include_top=False, weights="imagenet")

for layer in pre_trained_model.layers[:15]:
    layer.trainable = False

for layer in pre_trained_model.layers[15:]:
    layer.trainable = True

last_layer = pre_trained_model.get_layer('block5_pool')
last_output = last_layer.output

# Flatten the output layer to 1 dimension
x = GlobalMaxPooling2D()(last_output)
# Add a fully connected layer with 512 hidden units and ReLU activation
x = Dense(512, activation='relu')(x)
# Add a dropout rate of 0.5
x = Dropout(0.5)(x)
# Add a final sigmoid layer for classification
x = layers.Dense(2, activation='softmax')(x)

VGG16_model = Model(pre_trained_model.input,x)


VGG16_model.compile(loss='categorical_crossentropy',
              optimizer = optimizers.rmsprop(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])

VGG16_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applic
ations/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5 (https://stor
age.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_
ordering_tf_kernels_notop.h5)
58892288/58889256 [==============================] - 1s 0us/step

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-14-49bb1c047fce> in <module>()
     34
     35 VGG16_model.compile(loss='categorical_crossentropy',
---> 36               optimizer = optimizers.rmsprop(lr=0.0001, decay=1e-6),
     37               metrics=['accuracy'])
     38

AttributeError: module 'keras.optimizers' has no attribute 'rmsprop'
```

In [22]:

```
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Activation,Flatten,Dense,Dropout

model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(224,224,3)),
    MaxPooling2D(pool_size=(2,2)),


    Conv2D(64,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2,2)),

    Dropout(0.3),
    Flatten(),
#fully connected to get all relevant data
    Dense(128, activation='relu'),
#one more dropout for convergence' sake :)
    Dropout(0.5),
#output a softmax to squash the matrix into output probabilities
    Dense(2, activation='softmax')
])

model.compile(loss='categorical_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 109, 109, 64)      18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 52, 52, 64)        36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 26, 26, 64)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 26, 26, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 43264)             0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               5537920
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 258
=================================================================
Total params: 5,594,498
Trainable params: 5,594,498
```

```
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
!pip install -U efficientnet
```

In [ ]:

```
"""
import efficientnet.keras as efn
from keras import optimizers

base_model = efn.EfficientNetB0(input_shape = (224, 224, 3), include_top = False, weights =


x = model.output
x = Flatten()(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.5)(x)
predictions = Dense(1, activation="sigmoid")(x)
model_final = Model(input = model.input, output = predictions)

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

x = model.output
x = Flatten()(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.5)(x)
predictions = Dense(2, activation="softmax")(x)
model_final = Model(input = base_model.input, output = predictions)


model_final.compile(optimizers.rmsprop(lr=0.0001, decay=1e-6),loss='categorical_crossentrop
"""
```

Model Training

In [23]:

```
training = model.fit(Xtrain,Ytrain,validation_data=(Xtest,Ytest),epochs=20,verbose=1)
```

```
Epoch 1/20
39/39 [==============================] - 2s 51ms/step - loss: 0.7304 - acc
uracy: 0.5500 - val_loss: 0.5671 - val_accuracy: 0.6884
Epoch 2/20
39/39 [==============================] - 2s 45ms/step - loss: 0.5192 - acc
uracy: 0.7386 - val_loss: 0.6583 - val_accuracy: 0.6093
Epoch 3/20
39/39 [==============================] - 2s 45ms/step - loss: 0.4751 - acc
uracy: 0.7593 - val_loss: 0.5010 - val_accuracy: 0.7721
Epoch 4/20
39/39 [==============================] - 2s 45ms/step - loss: 0.3882 - acc
uracy: 0.8340 - val_loss: 0.4435 - val_accuracy: 0.7628
Epoch 5/20
39/39 [==============================] - 2s 45ms/step - loss: 0.2875 - acc
uracy: 0.8816 - val_loss: 0.3611 - val_accuracy: 0.8326
Epoch 6/20
39/39 [==============================] - 2s 45ms/step - loss: 0.2351 - acc
uracy: 0.9076 - val_loss: 0.2480 - val_accuracy: 0.9023
Epoch 7/20
39/39 [==============================] - 2s 46ms/step - loss: 0.1578 - acc
uracy: 0.9446 - val_loss: 0.1992 - val_accuracy: 0.9209
Epoch 8/20
39/39 [==============================] - 2s 46ms/step - loss: 0.1158 - acc
uracy: 0.9575 - val_loss: 0.2009 - val_accuracy: 0.9302
Epoch 9/20
39/39 [==============================] - 2s 45ms/step - loss: 0.1105 - acc
uracy: 0.9681 - val_loss: 0.2132 - val_accuracy: 0.9535
Epoch 10/20
39/39 [==============================] - 2s 46ms/step - loss: 0.0729 - acc
uracy: 0.9768 - val_loss: 0.1715 - val_accuracy: 0.9581
Epoch 11/20
39/39 [==============================] - 2s 46ms/step - loss: 0.0599 - acc
uracy: 0.9841 - val_loss: 0.1370 - val_accuracy: 0.9721
Epoch 12/20
39/39 [==============================] - 2s 46ms/step - loss: 0.0298 - acc
uracy: 0.9904 - val_loss: 0.1257 - val_accuracy: 0.9721
Epoch 13/20
39/39 [==============================] - 2s 46ms/step - loss: 0.0445 - acc
uracy: 0.9888 - val_loss: 0.1256 - val_accuracy: 0.9581
Epoch 14/20
39/39 [==============================] - 2s 45ms/step - loss: 0.0196 - acc
uracy: 0.9951 - val_loss: 0.1406 - val_accuracy: 0.9581
Epoch 15/20
39/39 [==============================] - 2s 45ms/step - loss: 0.0326 - acc
uracy: 0.9888 - val_loss: 0.1354 - val_accuracy: 0.9721
Epoch 16/20
39/39 [==============================] - 2s 46ms/step - loss: 0.0179 - acc
uracy: 0.9924 - val_loss: 0.1540 - val_accuracy: 0.9674
Epoch 17/20
39/39 [==============================] - 2s 46ms/step - loss: 0.0283 - acc
uracy: 0.9905 - val_loss: 0.2654 - val_accuracy: 0.9535
Epoch 18/20
39/39 [==============================] - 2s 46ms/step - loss: 0.0245 - acc
uracy: 0.9886 - val_loss: 0.1350 - val_accuracy: 0.9721
Epoch 19/20
39/39 [==============================] - 2s 46ms/step - loss: 0.0193 - acc
uracy: 0.9901 - val_loss: 0.1644 - val_accuracy: 0.9581
```

```
Epoch 20/20
39/39 [==============================] - 2s 45ms/step - loss: 0.0242 - acc
uracy: 0.9930 - val_loss: 0.1577 - val_accuracy: 0.9488
```

In [ ]:

```
training_VGG = VGG16_model.fit(Xtrain,Ytrain,validation_data=(Xtest,Ytest),epochs=20,verbos
```

Model Evaluation

In [24]:

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

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

```
7/7 [==============================] - 0s 20ms/step - loss: 0.1577 - accurac
y: 0.9488
test loss :  0.15773539245128632
test acc. : 0.9488372206687927
```

In [25]:

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

```
7/7 [==============================] - 0s 20ms/step
```

In [26]:

```
predictions
```

Out[26]:

```
array([[9.99970078e-01, 2.99280327e-05],
       [5.89273093e-08, 1.00000000e+00],
       [1.34115144e-17, 1.00000000e+00],
       [3.23254937e-07, 9.99999642e-01],
       [5.40808998e-09, 1.00000000e+00],
       [9.99999762e-01, 2.21617356e-07],
       [9.99927878e-01, 7.21437755e-05],
       [2.97578260e-12, 1.00000000e+00],
       [9.99400258e-01, 5.99756197e-04],
       [5.92136621e-01, 4.07863349e-01],
       [9.94073808e-01, 5.92619041e-03],
       [4.66255078e-05, 9.99953389e-01],
       [1.99780776e-03, 9.98002231e-01],
       [6.09787647e-03, 9.93902087e-01],
       [9.99862552e-01, 1.37398922e-04],
       [9.93861198e-01, 6.13884628e-03],
       [9.99897599e-01, 1.02377824e-04],
       [1.77190945e-04, 9.99822795e-01],
```

In [28]:

```
predicted_classes
```

Out[28]:

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

In [27]:

```
Ytest
```

Out[27]:

```
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 [29]:

```
Ytest_bin
```

Out[29]:

```
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 [30]:

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

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

```
0.9488372093023256
```

In [31]:

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

```
[[101   1]
 [ 10 103]]
```

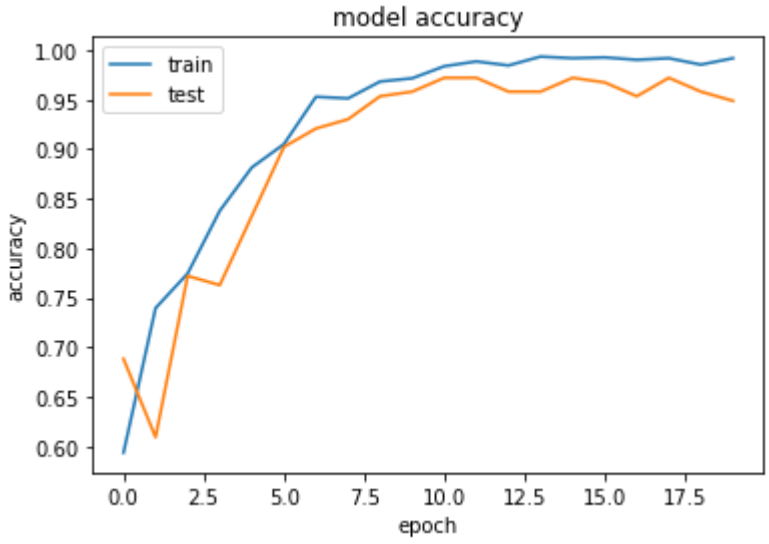
In [32]:

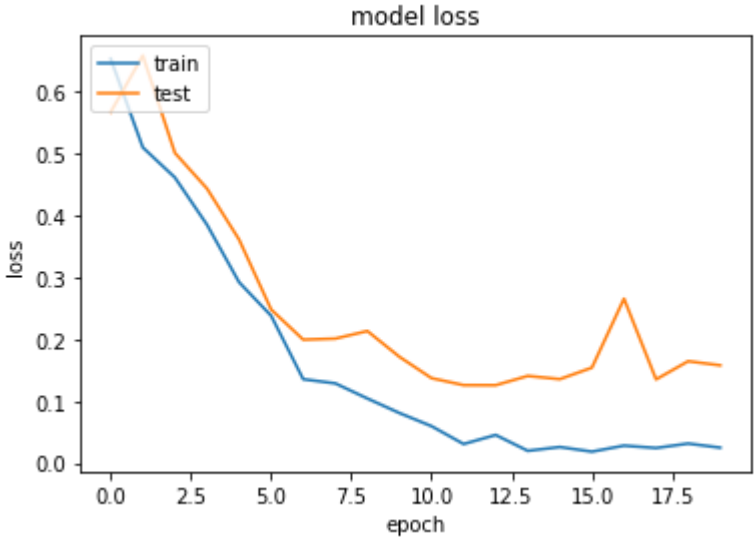
```
print(classification_report(Ytest_bin,predicted_classes,target_names=["yawn","no_yawn"]))
```

```
              precision    recall  f1-score   support

        yawn       0.91      0.99      0.95       102
     no_yawn       0.99      0.91      0.95       113

    accuracy                           0.95       215
   macro avg       0.95      0.95      0.95       215
weighted avg       0.95      0.95      0.95       215
```

In [33]:

```
import matplotlib.pyplot as plt

# summarize training for accuracy
plt.plot(training.history['accuracy'])
plt.plot(training.history['val_accuracy'])
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 [35]:

```
from sklearn.metrics import roc_curve,roc_auc_score

roc_auc_score(Ytest_bin,predicted_classes)
```

Out[35]:

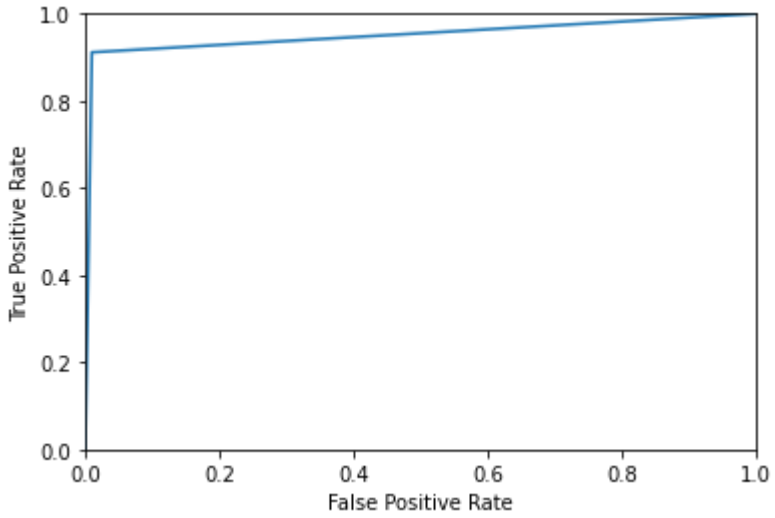
```
0.9508502516050669
```

In [36]:

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

In [37]:

```
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.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/")
"""
```