In [1]:
```
!git clone https://github.com/ksideks/UCSD.git
```

```
fatal: docelowa ścieżka „UCSD” już istnieje i nie jest pustym katalogiem.
```

In [2]:
```
!pip install keras-layer-normalization
```

```
Requirement already satisfied: keras-layer-normalization in ./jupyterenv/li
b/python3.8/site-packages (0.15.0)
Requirement already satisfied: Keras in ./jupyterenv/lib/python3.8/site-pac
kages (from keras-layer-normalization) (2.7.0)
Requirement already satisfied: numpy in ./jupyterenv/lib/python3.8/site-pac
kages (from keras-layer-normalization) (1.21.3)
```

In [3]:
```
TestVideoFile = {}
TestVideoFile[1] = range(59,152)
TestVideoFile[2] = range(49,175)
TestVideoFile[3] = range(90,200)
TestVideoFile[4] = range(30,168)
TestVideoFile[5] = list(range(4,90)) + list(range(139,200))
TestVideoFile[6] = list(range(0,100)) + list(range(109,200))
TestVideoFile[7] = range(0,175)
TestVideoFile[8] = range(0,94)
TestVideoFile[9] = range(0,48)
TestVideoFile[10] = range(0,140)
TestVideoFile[11] = range(69,165)
TestVideoFile[12] = range(130,200)
TestVideoFile[13] = range(0,156)
TestVideoFile[14] = range(6,200)
TestVideoFile[15] = range(137,200)
TestVideoFile[16] = range(122,200)
TestVideoFile[17] = range(0,47)
TestVideoFile[18] = range(53,120)
TestVideoFile[19] = range(63,138)
TestVideoFile[20] = range(44,175)
TestVideoFile[21] = range(30,200)
TestVideoFile[22] = range(16,107)
TestVideoFile[23] = range(8,165)
TestVideoFile[24] = range(49,171)
TestVideoFile[25] = range(39,135)
TestVideoFile[26] = range(77,144)
TestVideoFile[27] = range(9,122)
TestVideoFile[28] = range(104,200)
TestVideoFile[29] = list(range(0,15)) + list(range(44,113))
TestVideoFile[30] = range(174,200)
TestVideoFile[31] = range(0,180)
TestVideoFile[32] = list(range(0,52)) + list(range(64,115))
TestVideoFile[33] = range(4,165)
TestVideoFile[34] = range(0,121)
TestVideoFile[35] = range(85,200)
TestVideoFile[36] = range(14,108)
```

In [4]:
```
import os
os.environ["CUDA_VISIBLE_DEVICES"]="-1"
```

In [5]:

```
class Config:
  DATASET_PATH ="UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Train"
  TEST_PATH ="UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test"
  SINGLE_TEST_VIDEO_FILE = 1
  SINGLE_TEST_PATH = "UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test(
  BATCH_SIZE = 64
  EPOCHS = 50
  MODEL_PATH = "UCSD_v5/model_v8.hdf5"
  THRESHOLD = 0.95
```

In [6]:

```
from os import listdir
from os.path import isfile, join, isdir
from PIL import Image
import numpy as np
import shelve
def get_clips_by_stride(stride, frames_list, sequence_size):
    """ For data augmenting purposes.
    Parameters
    ----------
    stride : int
        The desired distance between two consecutive frames
    frames_list : list
        A list of sorted frames of shape 227 X 227
    sequence_size: int
        The size of the desired LSTM sequence
    Returns
    -------
    list
        A list of clips , 10 frames each
    """
    clips = []
    sz = len(frames_list)
    clip = np.zeros(shape=(sequence_size, 227, 227, 1))
    cnt = 0
    for start in range(0, stride):
        for i in range(start, sz, stride):
            clip[cnt, :, :, 0] = frames_list[i]
            cnt = cnt + 1
            if cnt == sequence_size:
                clips.append(np.copy(clip))
                cnt = 0
    return clips


def get_training_set():
    """
    Returns
    -------
    list
        A list of training sequences of shape (NUMBER_OF_SEQUENCES,SINGLE_S
    """
    ###################################
    # cache = shelve.open(Config.CACHE_PATH)
    # return cache["datasetLSTM"]
    ###################################
    clips = []
    # loop over the training folders (Train000,Train001,..)
    for f in sorted(listdir(Config.DATASET_PATH)):
        if isdir(join(Config.DATASET_PATH, f)):
            all_frames = []
            # loop over all the images in the folder (0.tif,1.tif,..,199.t
            for c in sorted(listdir(join(Config.DATASET_PATH, f))):
                if str(join(join(Config.DATASET_PATH, f), c))[-3:] == "tif
                    img = Image.open(join(join(Config.DATASET_PATH, f), c)
                    img = np.array(img, dtype=np.float32) / 256.0
                    all_frames.append(img)
            # get the 10-frames sequences from the list of images after ap
            for stride in range(1, 3):
                clips.extend(get_clips_by_stride(stride=stride, frames_lis
    return clips
```

In [7]:

```
import keras
import tensorflow as tf
from keras.layers import Conv2DTranspose, ConvLSTM2D, BatchNormalization, T
from keras.models import Sequential, load_model
def get_model(reload_model=True):
    """
    Parameters
    ----------
    reload_model : bool
        Load saved model or retrain it
    """
    if not reload_model:
        return load_model(Config.MODEL_PATH,custom_objects={'LayerNormaliza
    training_set = get_training_set()
    training_set = np.array(training_set)
    training_set = training_set.reshape(-1,10,227,227,1)

    seq = Sequential()

    '''
    seq.add(TimeDistributed(Conv2D(128, (11, 11), strides=4, padding="same"
    seq.add(LayerNormalization())
    seq.add(TimeDistributed(Conv2D(64, (5, 5), strides=2, padding="same"))
    seq.add(LayerNormalization())
    # # # # #
    seq.add(ConvLSTM2D(64, (3, 3), padding="same", return_sequences=True))
    seq.add(LayerNormalization())
    seq.add(ConvLSTM2D(32, (3, 3), padding="same", return_sequences=True))
    seq.add(LayerNormalization())
    seq.add(ConvLSTM2D(64, (3, 3), padding="same", return_sequences=True))
    seq.add(LayerNormalization())
    # # # # #
    seq.add(TimeDistributed(Conv2DTranspose(64, (5, 5), strides=2, padding=
    seq.add(LayerNormalization())
    seq.add(TimeDistributed(Conv2DTranspose(128, (11, 11), strides=4, paddi
    seq.add(LayerNormalization())
    seq.add(TimeDistributed(Conv2D(1, (11, 11), activation="sigmoid", paddi

    print(seq.summary())

    seq.compile(loss='mse', optimizer=tf.keras.optimizers.Adam(lr=1e-4, de
    '''

    #AUTOENCODER --> spatial part

    seq.add(TimeDistributed(Conv2D(128, (11, 11), strides=4, padding="vali
    seq.add(LayerNormalization())
    seq.add(TimeDistributed(Conv2D(64, (5, 5), strides=2, padding="valid",
    seq.add(LayerNormalization())


    # Convolutional Long-short term memory --> temporal part
    seq.add(ConvLSTM2D(64, (3, 3), strides=1, padding="same", return_seque
    seq.add(LayerNormalization())
    seq.add(ConvLSTM2D(32, (3, 3), strides=1, padding="same", return_seque
    seq.add(LayerNormalization())
    seq.add(ConvLSTM2D(64, (3, 3), strides=1, padding="same", return_seque
    seq.add(LayerNormalization())

    # AUTODECODER --> spatial part

    seq.add(TimeDistributed(Conv2DTranspose(128, (5, 5), strides=2, paddin
```

```
    seq.compile(loss= 'mse' , optimizer=tf.keras.optimizers.Adam(lr=1e-3)) #
    seq.fit(training_set, training_set,
            batch_size=Config.BATCH_SIZE, epochs=Config.EPOCHS, shuffle=Fa
    seq.save(Config.MODEL_PATH)
    return seq
```

```
2021-11-11 07:33:50.398256: W tensorflow/stream_executor/platform/default/d
so_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerro
r: libcudart.so.11.0: cannot open shared object file: No such file or direc
tory
2021-11-11 07:33:50.398329: I tensorflow/stream_executor/cuda/cudart_stub.c
c:29] Ignore above cudart dlerror if you do not have a GPU set up on your m
achine.
```

In [8]:

```
def get_single_test():
    sz = 200
    test = np.zeros(shape=(sz, 227, 227, 1))
    cnt = 0
    for f in sorted(listdir(Config.SINGLE_TEST_PATH)):
        if str(join(Config.SINGLE_TEST_PATH, f))[-3:] == "tif":
            img = Image.open(join(Config.SINGLE_TEST_PATH, f)).resize((227
            img = np.array(img, dtype=np.float32) / 256.0
            test[cnt, :, :, 0] = img
            cnt = cnt + 1
    return test
```

In [9]:

```
import matplotlib.pyplot as plt
import pandas as pd

def evaluate(reload_model=False):
    model = get_model(reload_model)
    print("got model")
    test = get_single_test()
    print(test.shape)
    sz = test.shape[0] - 10 + 1
    sequences = np.zeros((sz, 10, 227, 227, 1))
    # apply the sliding window technique to get the sequences
    for i in range(0, sz):
        clip = np.zeros((10, 227, 227, 1))
        for j in range(0, 10):
            clip[j] = test[i + j, :, :, :]
        sequences[i] = clip

    print("got data")
    # get the reconstruction cost of all the sequences
    reconstructed_sequences = model.predict(sequences,batch_size=4)
    sequences_reconstruction_cost = np.array([np.linalg.norm(np.subtract(se
    sa = (sequences_reconstruction_cost - np.min(sequences_reconstruction_c
    sr = 1.0 - sa

    # plot the regularity scores
    plt.plot(sr)
    plt.ylabel('regularity score Sr(t)')
    plt.xlabel('frame t')
    plt.show()

    return sr, sequences
```

In [10]:

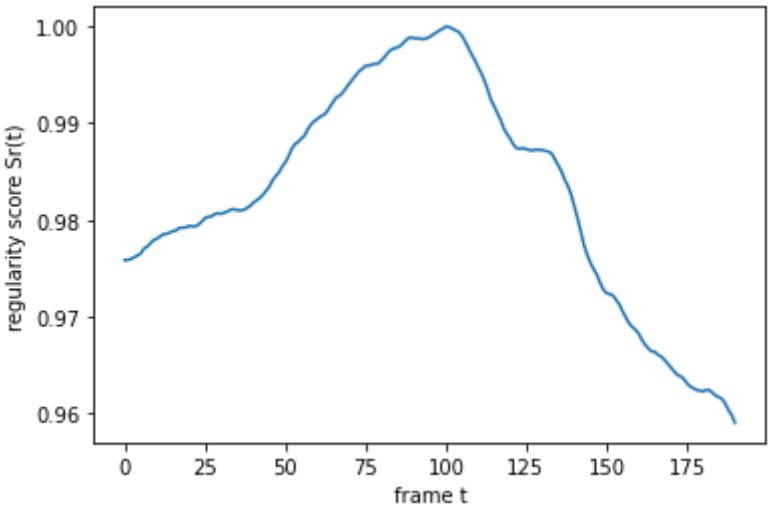
```
pr, before_reconstuction = evaluate(reload_model=True)
```

```
2021-11-11 07:34:07.569166: W tensorflow/stream_executor/platform/default/d
so_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: li
bcuda.so.1: cannot open shared object file: No such file or directory
2021-11-11 07:34:07.569222: W tensorflow/stream_executor/cuda/cuda_driver.c
c:269] failed call to cuInit: UNKNOWN ERROR (303)
2021-11-11 07:34:07.569246: I tensorflow/stream_executor/cuda/cuda_diagnost
ics.cc:156] kernel driver does not appear to be running on this host (ml):
/proc/driver/nvidia/version does not exist
2021-11-11 07:34:07.569522: I tensorflow/core/platform/cpu_feature_guard.c
c:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network
Library (oneDNN) to use the following CPU instructions in performance-criti
cal operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed (TimeDistr  (None, 10, 55, 55, 128)  15616
 ibuted)

 layer_normalization (LayerN  (None, 10, 55, 55, 128)  256
 ormalization)

 time_distributed_1 (TimeDis  (None, 10, 26, 26, 64)   204864
 tributed)
```

```
 layer_normalization_1 (Laye  (None, 10, 26, 26, 64)   128
 rNormalization)

 conv_lstm2d (ConvLSTM2D)    (None, 10, 26, 26, 64)    295168

 layer_normalization_2 (Laye  (None, 10, 26, 26, 64)   128
 rNormalization)

 conv_lstm2d_1 (ConvLSTM2D)  (None, 10, 26, 26, 32)    110720

 layer_normalization_3 (Laye  (None, 10, 26, 26, 32)   64
 rNormalization)

 conv_lstm2d_2 (ConvLSTM2D)  (None, 10, 26, 26, 64)    221440

 layer_normalization_4 (Laye  (None, 10, 26, 26, 64)   128
 rNormalization)

 time_distributed_2 (TimeDis  (None, 10, 55, 55, 128)  204928
 tributed)

 layer_normalization_5 (Laye  (None, 10, 55, 55, 128)  256
 rNormalization)

 time_distributed_3 (TimeDis  (None, 10, 227, 227, 1)  15489
 tributed)

 layer_normalization_6 (Laye  (None, 10, 227, 227, 1)  2
 rNormalization)

=================================================================
Total params: 1,069,187
Trainable params: 1,069,187
Non-trainable params: 0

_________________________________________________________________
None
/home/user/notebook/jupyterenv/lib/python3.8/site-packages/keras/optimizer_
v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning
_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
Epoch 1/50
22/22 [==============================] - 487s 21s/step - loss: 0.1731
Epoch 2/50
22/22 [==============================] - 444s 20s/step - loss: 0.1578
Epoch 3/50
22/22 [==============================] - 448s 20s/step - loss: 0.1438
Epoch 4/50
22/22 [==============================] - 479s 22s/step - loss: 0.1310
Epoch 5/50
22/22 [==============================] - 501s 23s/step - loss: 0.1195
Epoch 6/50
22/22 [==============================] - 490s 22s/step - loss: 0.1092
Epoch 7/50
22/22 [==============================] - 484s 22s/step - loss: 0.0999
Epoch 8/50
22/22 [==============================] - 494s 22s/step - loss: 0.0917
Epoch 9/50
22/22 [==============================] - 482s 22s/step - loss: 0.0844
Epoch 10/50
22/22 [==============================] - 479s 22s/step - loss: 0.0780
Epoch 11/50
22/22 [==============================] - 486s 22s/step - loss: 0.0723
```

```
Epoch 12/50
22/22 [==============================] - 480s 22s/step - loss: 0.0674
Epoch 13/50
22/22 [==============================] - 482s 22s/step - loss: 0.0631
Epoch 14/50
22/22 [==============================] - 492s 22s/step - loss: 0.0595
Epoch 15/50
22/22 [==============================] - 479s 22s/step - loss: 0.0563
Epoch 16/50
22/22 [==============================] - 483s 22s/step - loss: 0.0536
Epoch 17/50
22/22 [==============================] - 490s 22s/step - loss: 0.0513
Epoch 18/50
22/22 [==============================] - 476s 22s/step - loss: 0.0494
Epoch 19/50
22/22 [==============================] - 482s 22s/step - loss: 0.0478
Epoch 20/50
22/22 [==============================] - 486s 22s/step - loss: 0.0464
Epoch 21/50
22/22 [==============================] - 471s 21s/step - loss: 0.0453
Epoch 22/50
22/22 [==============================] - 486s 22s/step - loss: 0.0444
Epoch 23/50
22/22 [==============================] - 489s 22s/step - loss: 0.0437
Epoch 24/50
22/22 [==============================] - 472s 21s/step - loss: 0.0431
Epoch 25/50
22/22 [==============================] - 486s 22s/step - loss: 0.0426
Epoch 26/50
22/22 [==============================] - 487s 22s/step - loss: 0.0422
Epoch 27/50
22/22 [==============================] - 471s 21s/step - loss: 0.0419
Epoch 28/50
22/22 [==============================] - 483s 22s/step - loss: 0.0417
Epoch 29/50
22/22 [==============================] - 481s 22s/step - loss: 0.0415
Epoch 30/50
22/22 [==============================] - 473s 21s/step - loss: 0.0414
Epoch 31/50
22/22 [==============================] - 488s 22s/step - loss: 0.0413
Epoch 32/50
22/22 [==============================] - 478s 22s/step - loss: 0.0412
Epoch 33/50
22/22 [==============================] - 478s 22s/step - loss: 0.0411
Epoch 34/50
22/22 [==============================] - 485s 22s/step - loss: 0.0411
Epoch 35/50
22/22 [==============================] - 478s 22s/step - loss: 0.0411
Epoch 36/50
22/22 [==============================] - 475s 22s/step - loss: 0.0410
Epoch 37/50
22/22 [==============================] - 474s 22s/step - loss: 0.0410
Epoch 38/50
22/22 [==============================] - 474s 22s/step - loss: 0.0410
Epoch 39/50
22/22 [==============================] - 468s 21s/step - loss: 0.0410
Epoch 40/50
22/22 [==============================] - 482s 22s/step - loss: 0.0410
Epoch 41/50
22/22 [==============================] - 487s 22s/step - loss: 0.0410
Epoch 42/50
22/22 [==============================] - 470s 21s/step - loss: 0.0410
Epoch 43/50
```

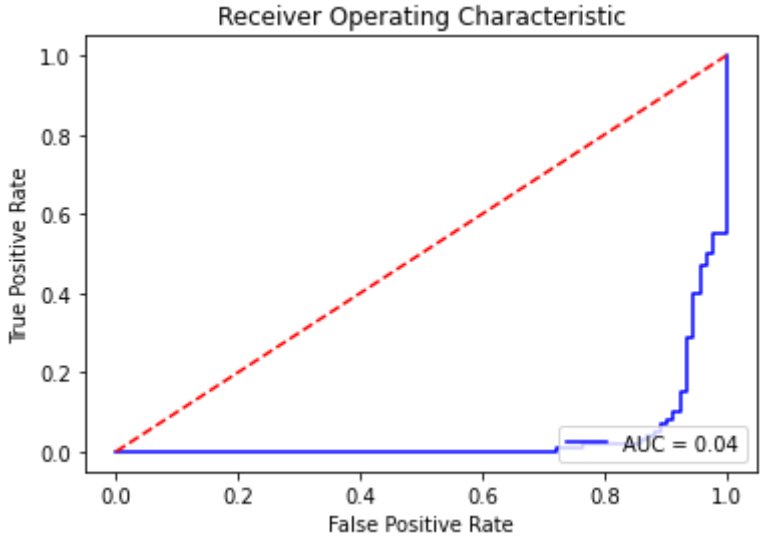
```
22/22 [==============================] - 484s 22s/step - loss: 0.0410
Epoch 44/50
22/22 [==============================] - 486s 22s/step - loss: 0.0410
Epoch 45/50
22/22 [==============================] - 467s 21s/step - loss: 0.0410
Epoch 46/50
22/22 [==============================] - 481s 22s/step - loss: 0.0410
Epoch 47/50
22/22 [==============================] - 478s 22s/step - loss: 0.0410
Epoch 48/50
22/22 [==============================] - 470s 21s/step - loss: 0.0410
Epoch 49/50
22/22 [==============================] - 481s 22s/step - loss: 0.0410
Epoch 50/50
22/22 [==============================] - 476s 22s/step - loss: 0.0410
got model
(200, 227, 227, 1)
got data
```

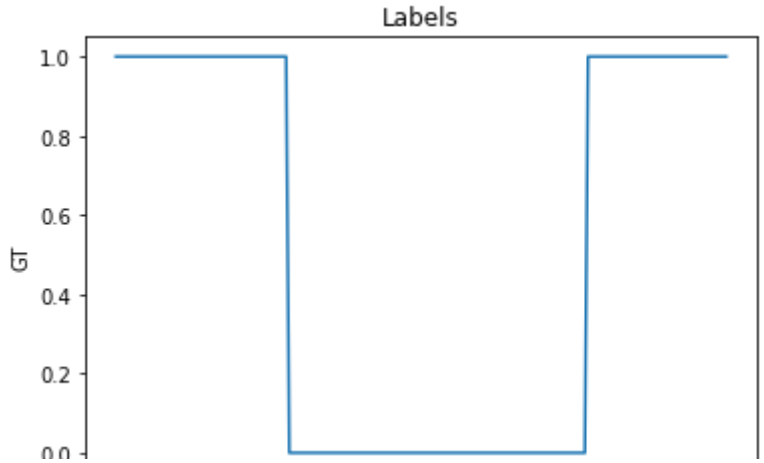


In [11]:

```
from sklearn import metrics

def plotROC(pr):
  y_pred = pr
  y_test = [1 for element in range(0, 200)]

  for i in TestVideoFile[Config.SINGLE_TEST_VIDEO_FILE]:
    y_test[i] = 0

  #wariant 1
  # y_test = y_test[9:]
  #wariant 2
  #y_test = y_test[:191]
  #wariant 3
  y_test = y_test[5:196]

  fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred)
  fnr = 1 - tpr
  auc = metrics.roc_auc_score(y_test, y_pred)

  eer_threshold = thresholds[np.nanargmin(np.absolute((fnr - fpr)))]
  eer = fpr[np.nanargmin(np.absolute((fnr - fpr)))]

  optimal = np.argmax(tpr - fpr)
  optimal_threshold = thresholds[optimal]


  #print("FPR: ", fpr)
  #print("TPR: ", tpr)
  #print("THRESHOLDS", thresholds)
  print("AUC: ", auc)
  print("EER: ", eer)
  print("EER THRESHOLD: ", eer_threshold)
  print("Optimal threshold value is:", optimal_threshold)

  plt.title('Receiver Operating Characteristic')
  plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % auc)
  plt.legend(loc = 'lower right')
  plt.plot([0, 1], [0, 1],'r--')
  plt.ylabel('True Positive Rate')
  plt.xlabel('False Positive Rate')
  plt.show()

  plt.plot(y_test)
  plt.title('Labels')
  plt.ylabel('GT')
  plt.xlabel('Frame')
  plt.show()

  return auc, eer

plotROC(pr)
```

```
AUC:  0.037744129910028526
EER:  0.9139784946236559
EER THRESHOLD:  0.9835137571629116
Optimal threshold value is: 2.0
```

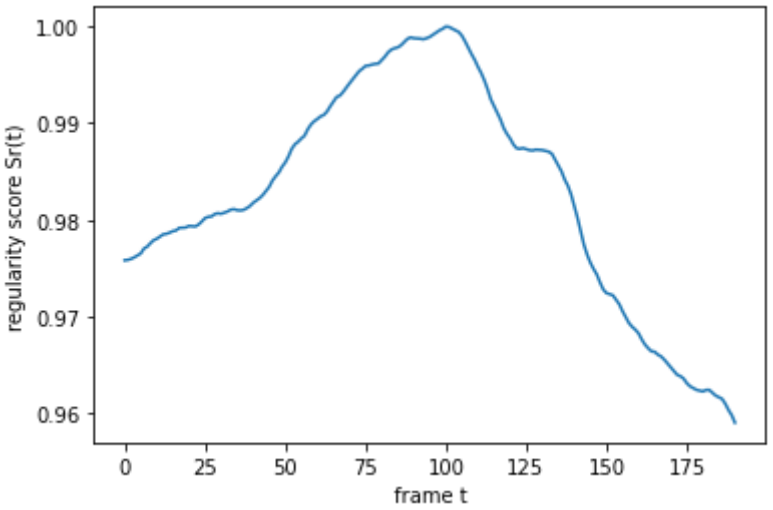

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
AUC = 0.04
0.0
0.2
0.4
0.6
0.8
1.0


Labels
GT
0.0
0.2
0.4
0.6
0.8
1.0

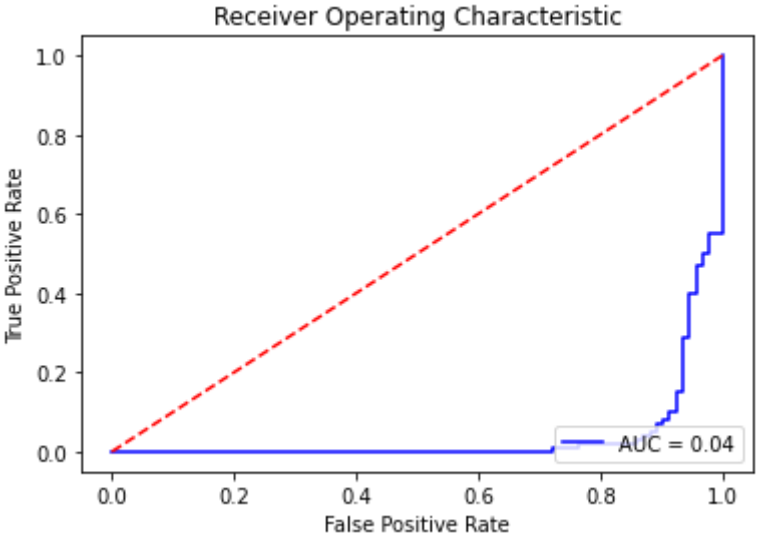
In [12]:

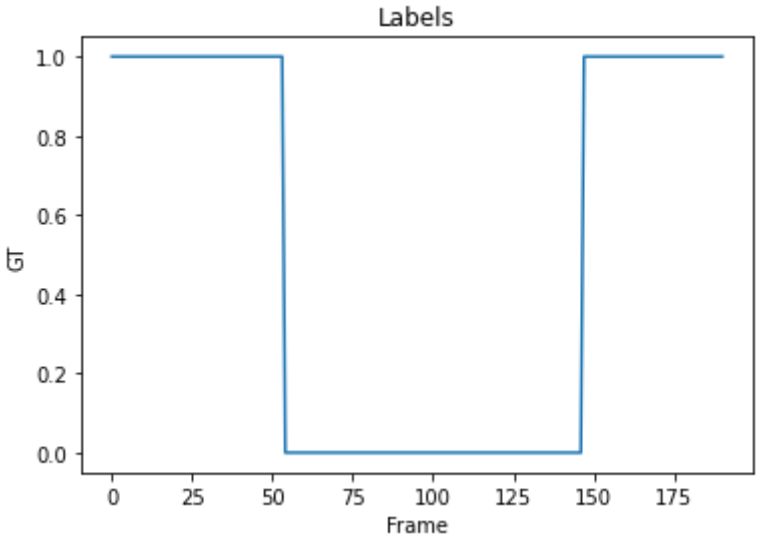
```
from os import listdir
from os.path import isfile, join, isdir


clips = []
# loop over the training folders (Train000,Train001,..)
for f in sorted(listdir(Config.TEST_PATH)):
    if isdir(join(Config.TEST_PATH, f)):
      if not 'gt' in f:
        clips.append(join(Config.TEST_PATH, f))


scores = []

for i in range(len(clips)):
  if(i == 16): #skip clip 17
    continue

  Config.SINGLE_TEST_PATH = clips[i]
  Config.SINGLE_TEST_VIDEO_FILE = i+1

  print("PATH: ", Config.SINGLE_TEST_PATH)
  print("GT: ", Config.SINGLE_TEST_VIDEO_FILE)

  pr, before_reconstuction = evaluate()
  scores.append(plotROC(pr))


mean = np.mean(scores, axis=0)
#print(scores)
print("AUC: ", mean[0])
print("EER: ", mean[1])
```

```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test001
GT:  1
got model
(200, 227, 227, 1)
got data
```

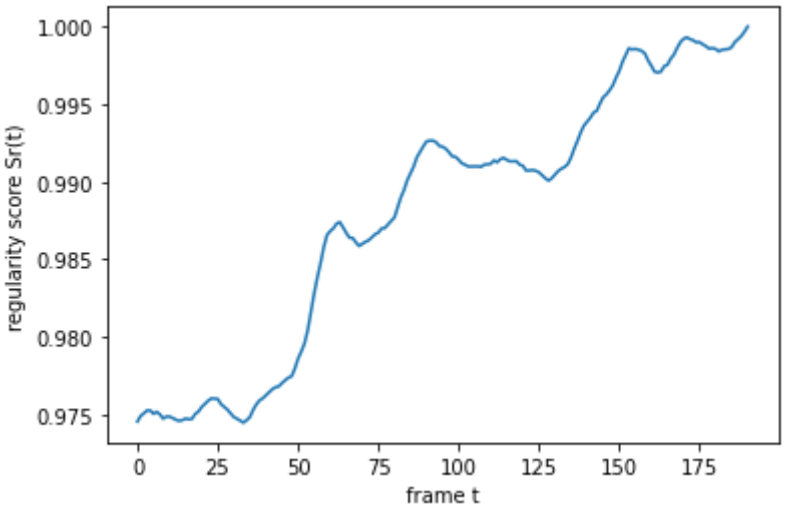



```
AUC:  0.037744129910028526
EER:  0.9139784946236559
EER THRESHOLD:  0.9835137571629116
Optimal threshold value is: 2.0
```

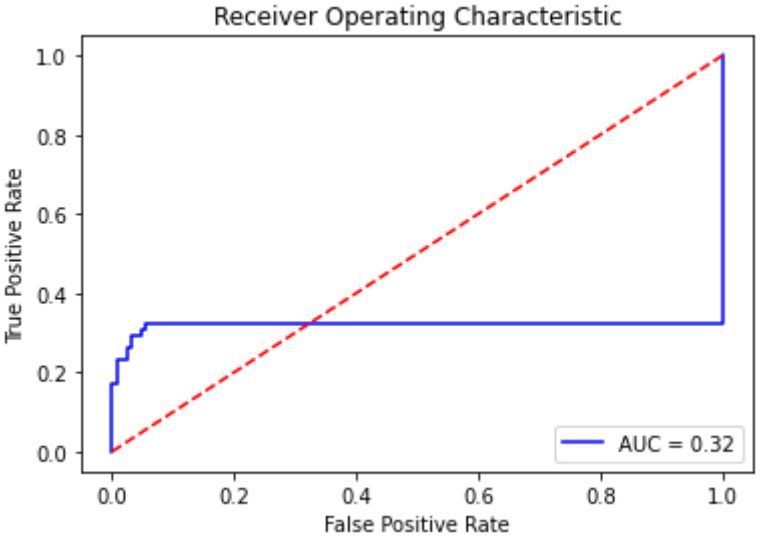


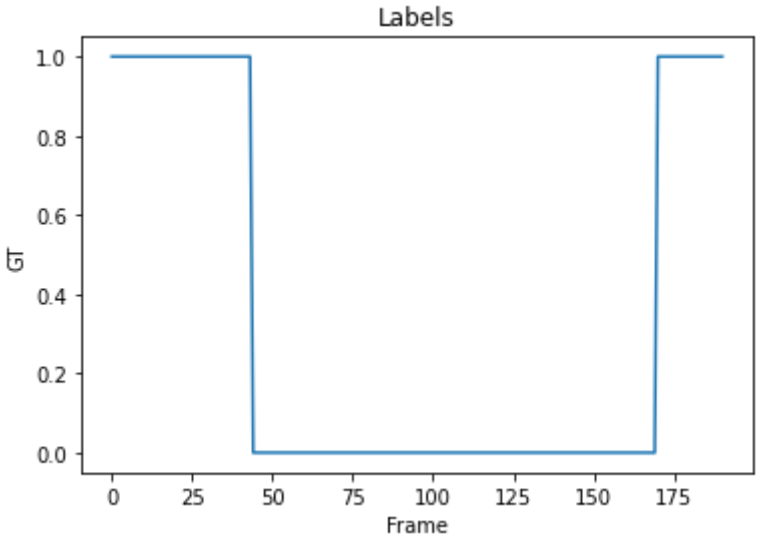



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test002
GT:  2
got model
(200, 227, 227, 1)
got data
```

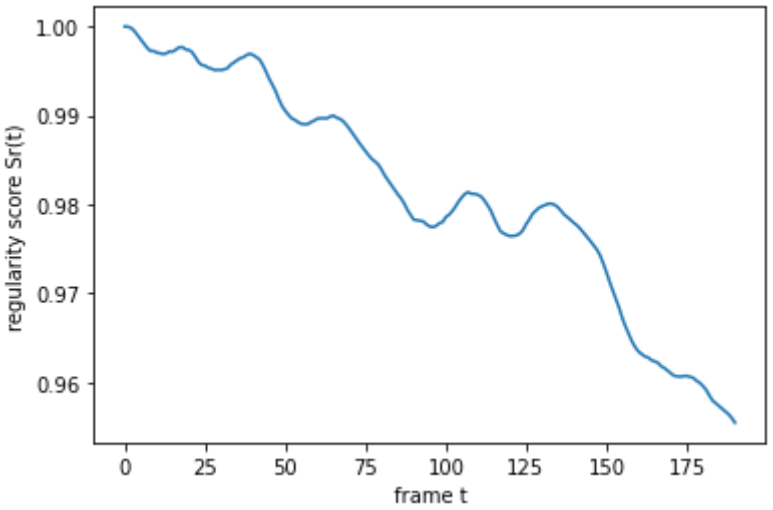



```
AUC:  0.3192918192918193
EER:  1.0
EER THRESHOLD:  0.9768251632191567
Optimal threshold value is: 0.9984002336080328
```

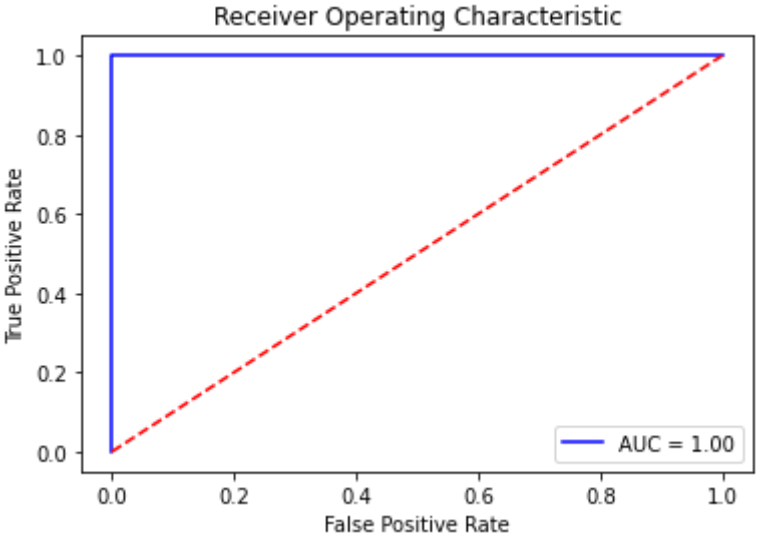


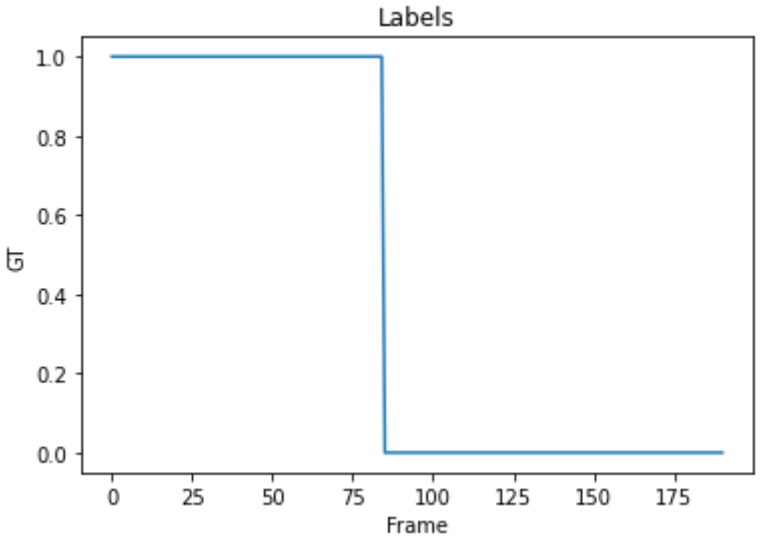



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test003
GT:  3
got model
(200, 227, 227, 1)
got data
```

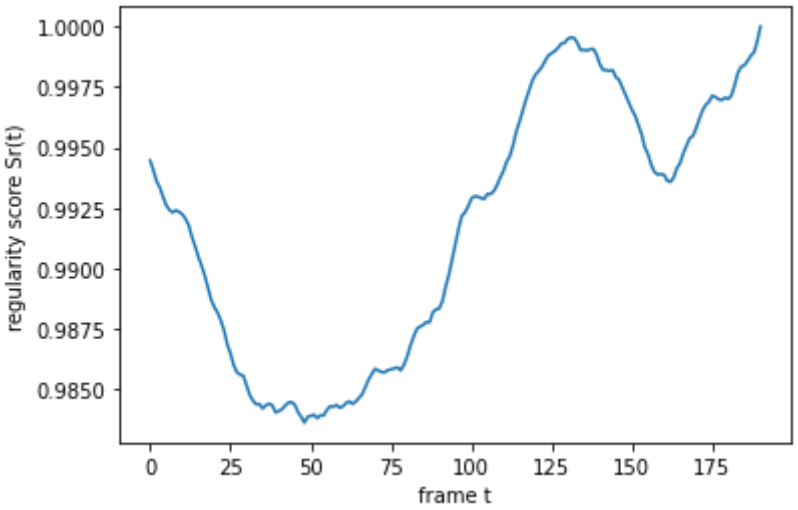



```
AUC:  1.0
EER:  0.0
EER THRESHOLD:  0.9816749186282272
Optimal threshold value is: 0.9816749186282272
```

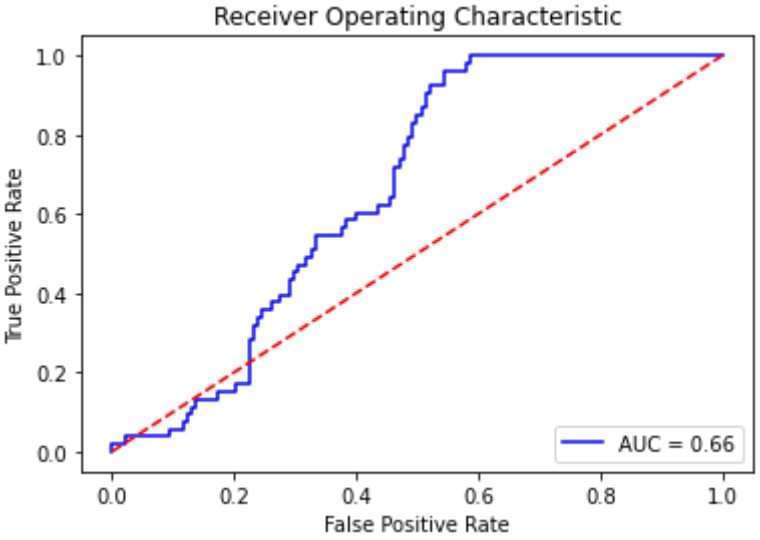


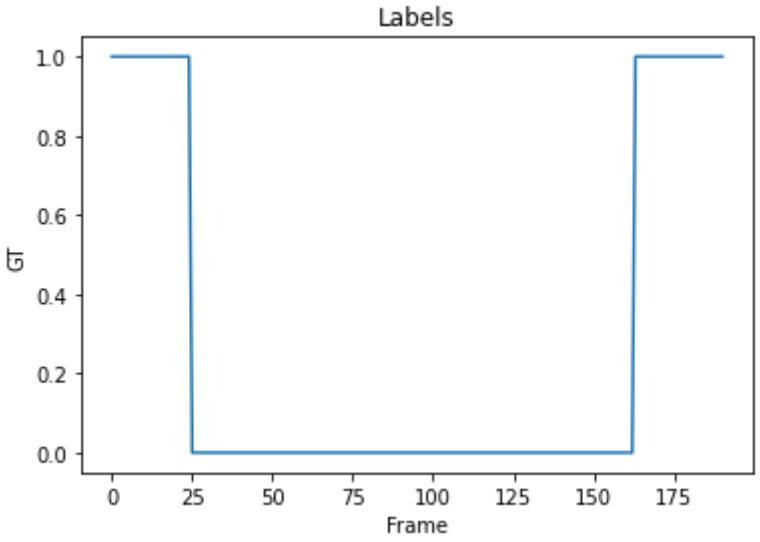



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test004
GT:  4
got model
(200, 227, 227, 1)
got data
```

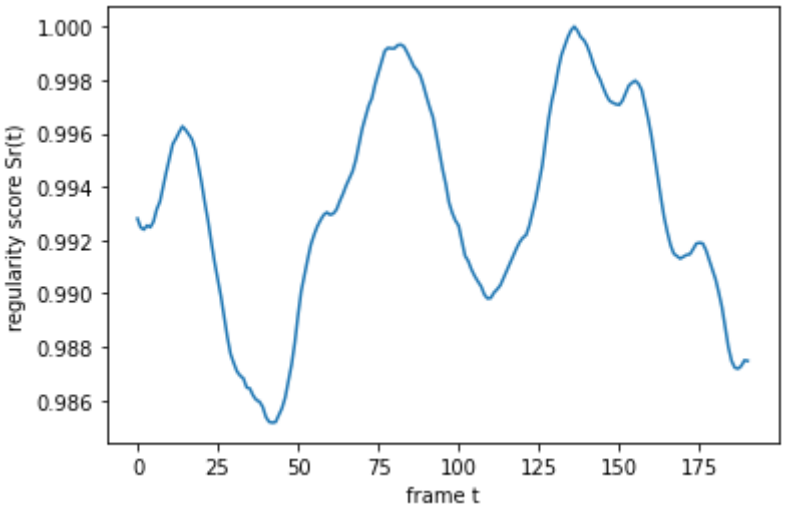



```
AUC:  0.6614711512168444
EER:  0.39855072463768115
EER THRESHOLD:  0.9933429227279764
Optimal threshold value is: 0.9878503803806926
```

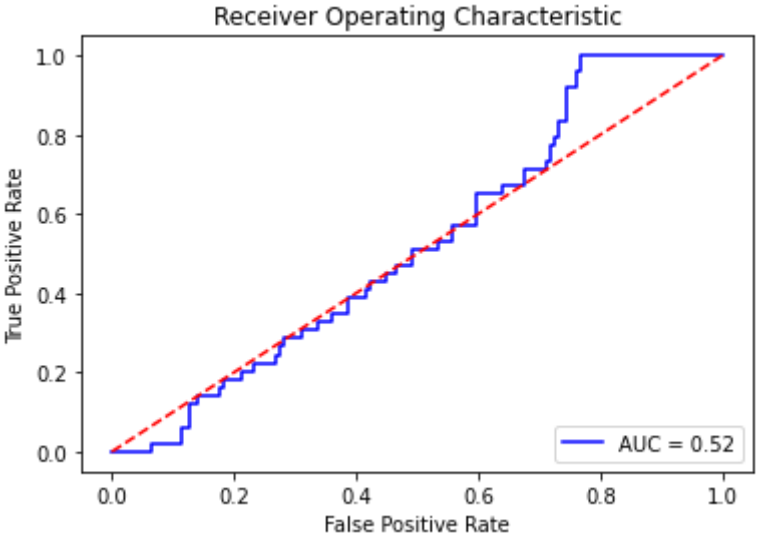


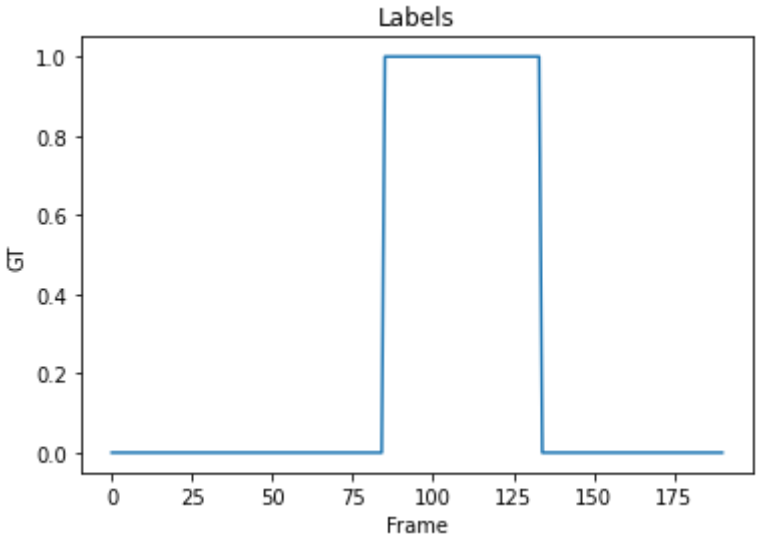



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test005
GT:  5
got model
(200, 227, 227, 1)
got data
```

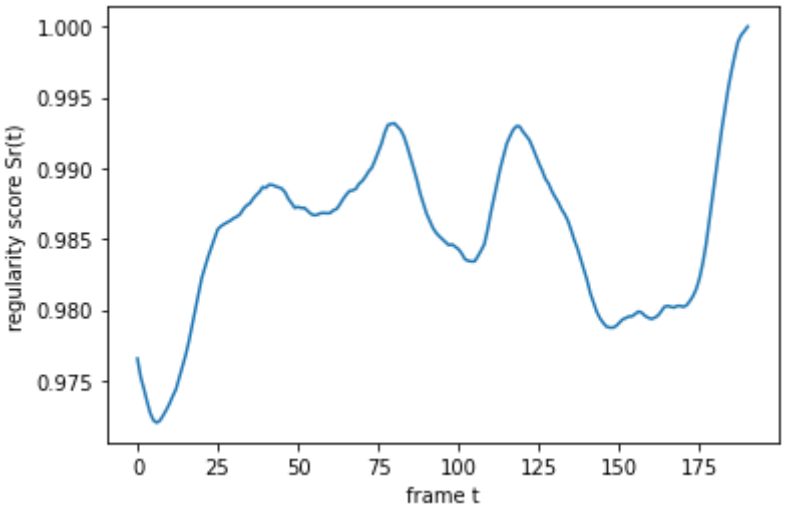



```
AUC:  0.522995113538373
EER:  0.49295774647887325
EER THRESHOLD:  0.9930222278281151
Optimal threshold value is: 0.9898115965823612
```

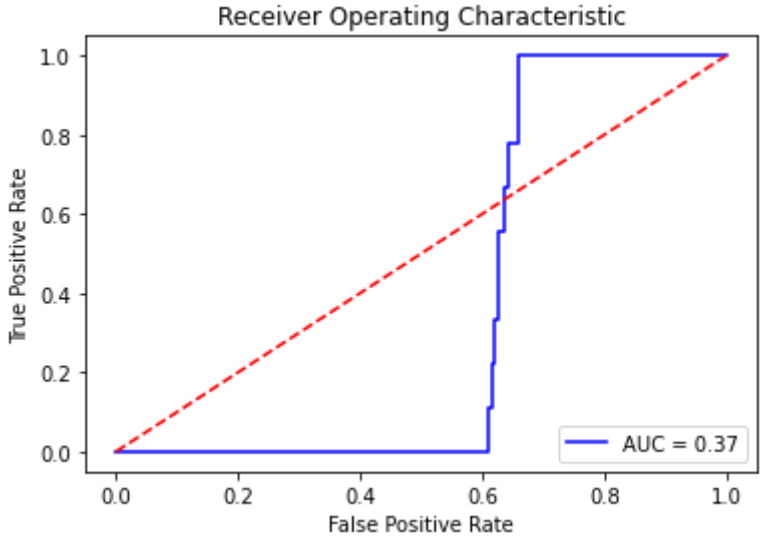


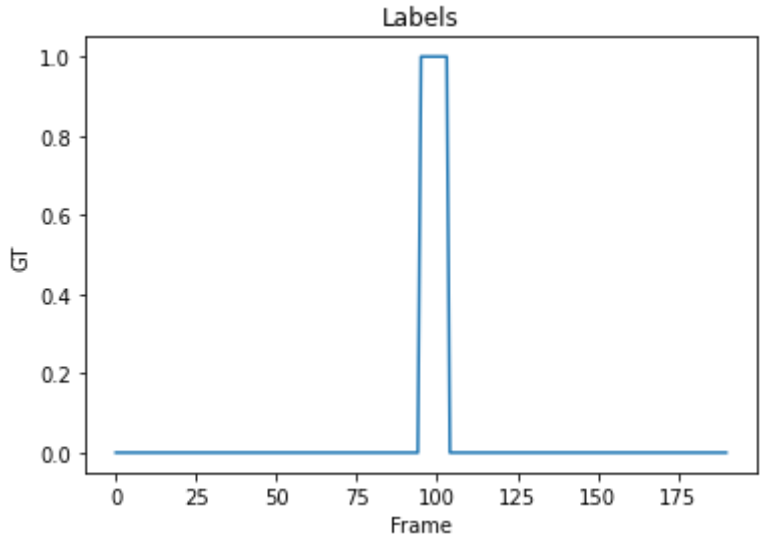



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test006
GT:  6
got model
(200, 227, 227, 1)
got data
```

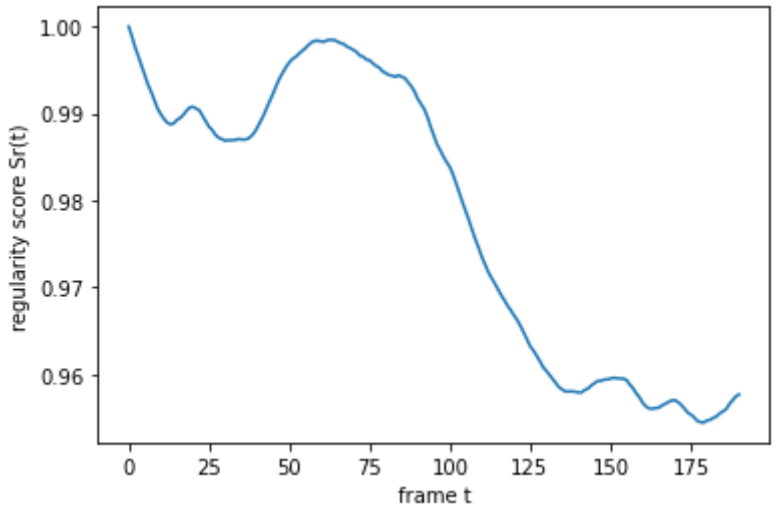



```
AUC:  0.3669108669108669
EER:  0.6263736263736264
EER THRESHOLD:  0.9845622548711405
Optimal threshold value is: 0.9834573287544133
```

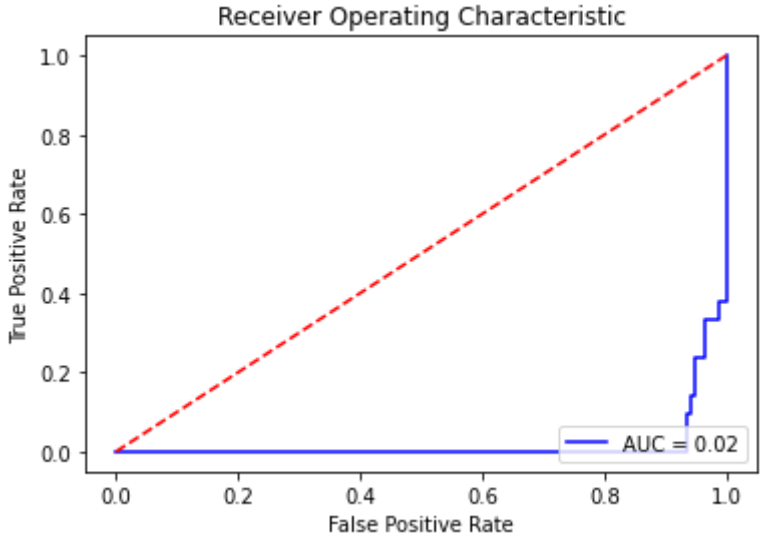


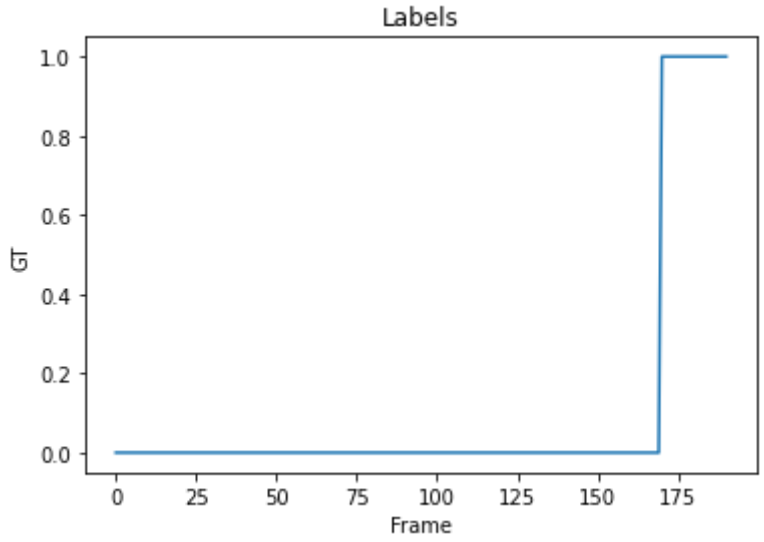



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test007
GT:  7
got model
(200, 227, 227, 1)
got data
```

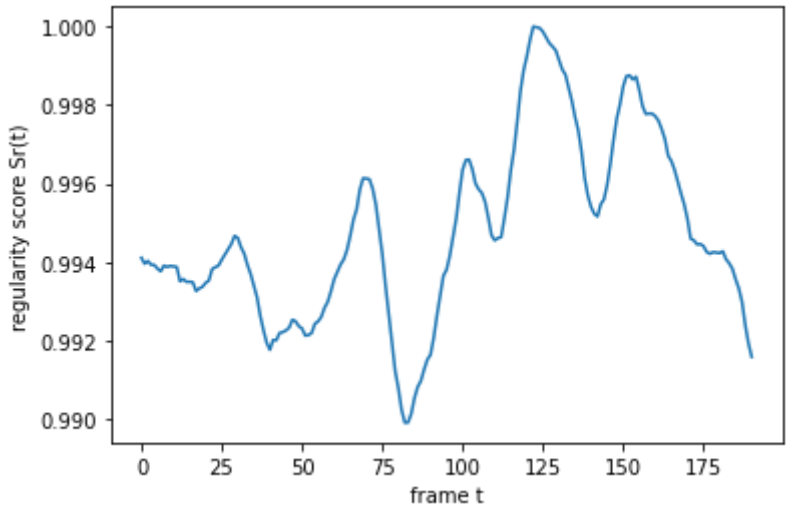



```
AUC:  0.017927170868347338
EER:  0.9352941176470588
EER THRESHOLD:  0.9574221568355827
Optimal threshold value is: 2.0
```

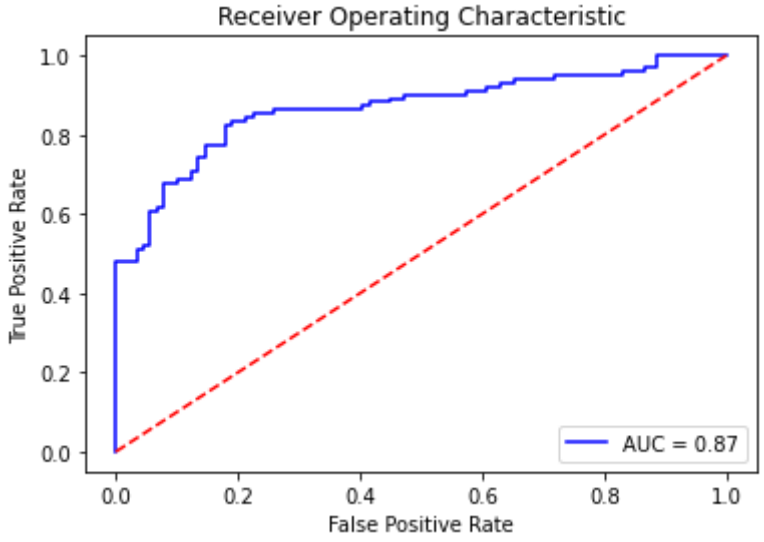


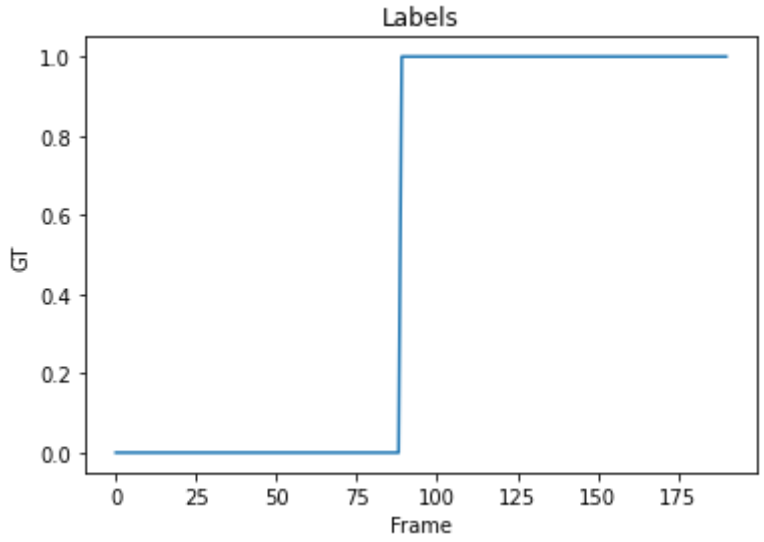



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test008
GT:  8
got model
(200, 227, 227, 1)
got data
```

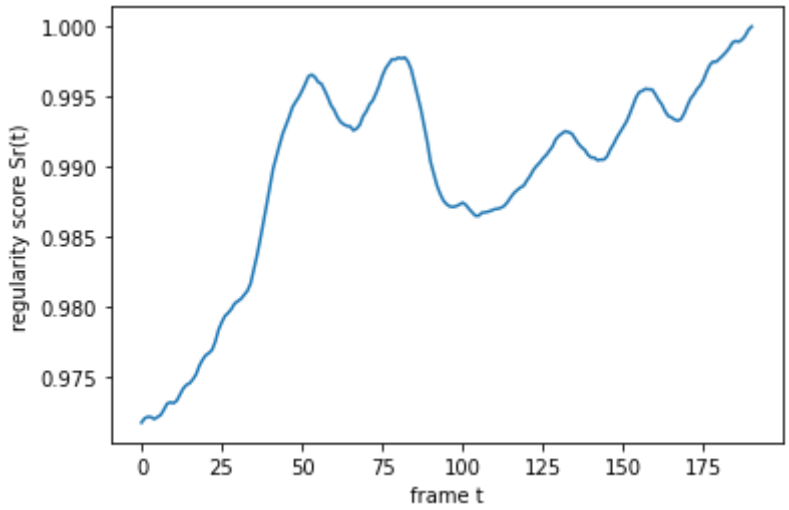



```
AUC:  0.8668208856576338
EER:  0.1797752808988764
EER THRESHOLD:  0.9942292192235466
Optimal threshold value is: 0.9942292192235466
```

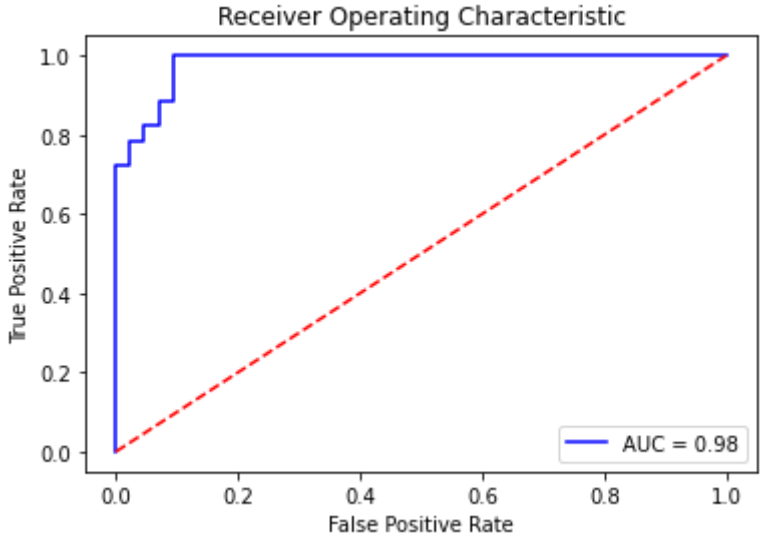


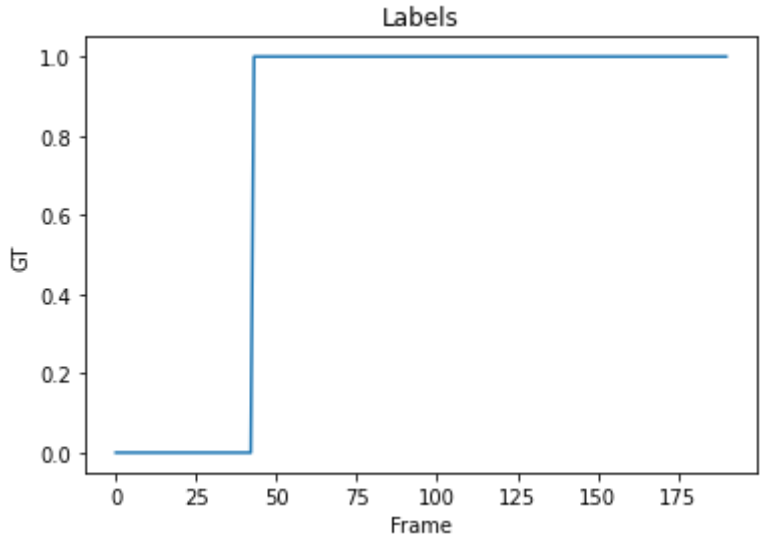



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test009
GT:  9
got model
(200, 227, 227, 1)
got data
```

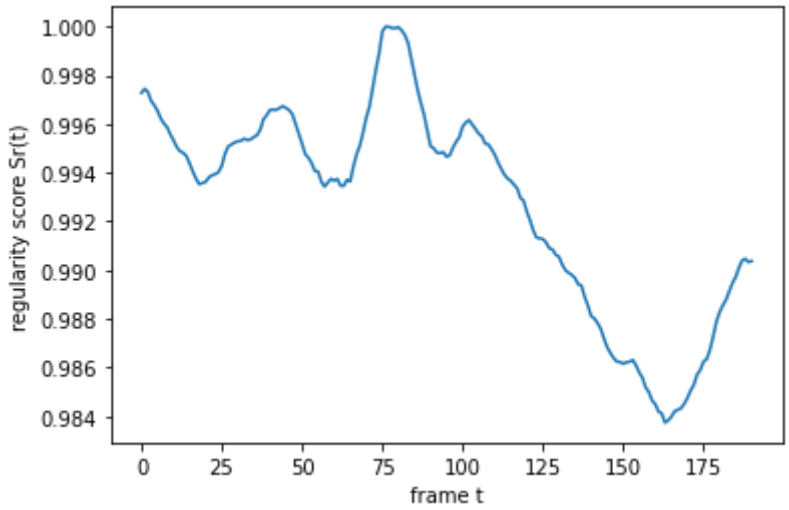



```
AUC:  0.9817724701445633
EER:  0.09302325581395349
EER THRESHOLD:  0.9872809193185624
Optimal threshold value is: 0.9864660993823523
```

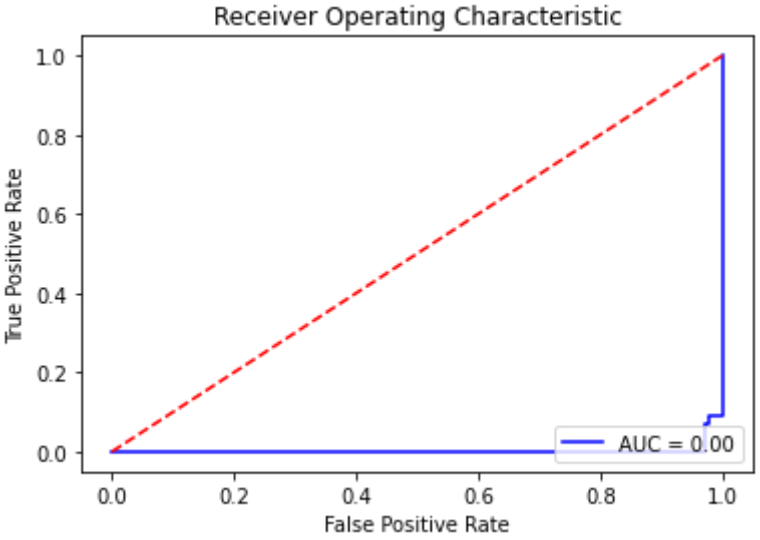


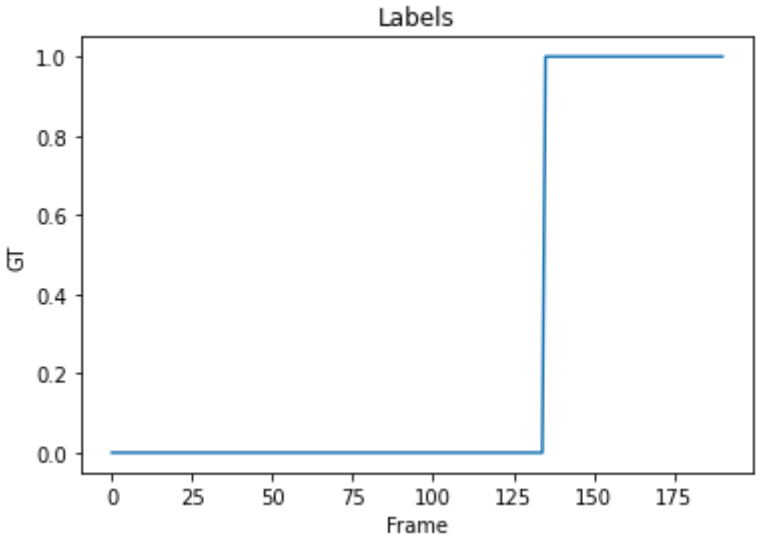



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test010
GT:  10
got model
(200, 227, 227, 1)
got data
```

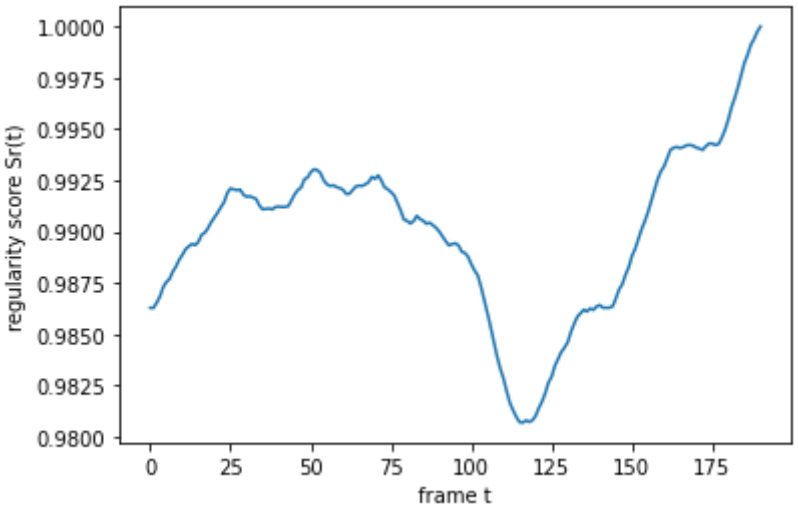



```
AUC:  0.0025132275132275167
EER:  0.9703703703703703
EER THRESHOLD:  0.9905462429247556
Optimal threshold value is: 2.0
```

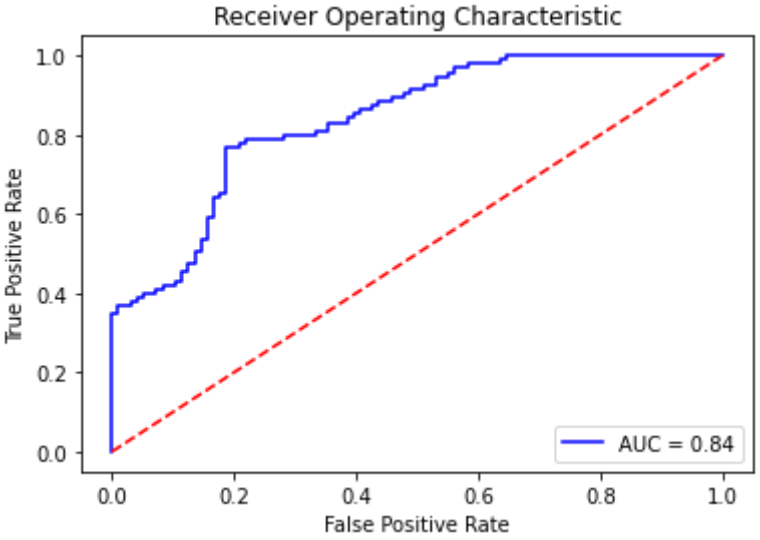


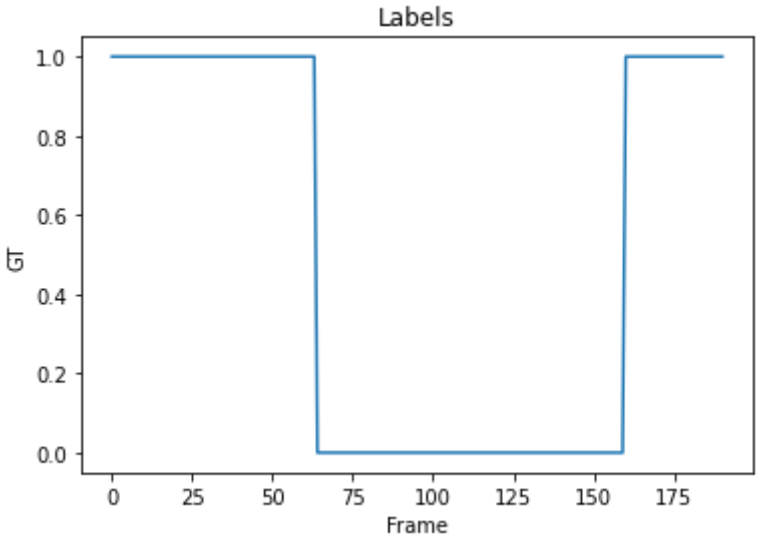



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test011
GT:  11
got model
(200, 227, 227, 1)
got data
```

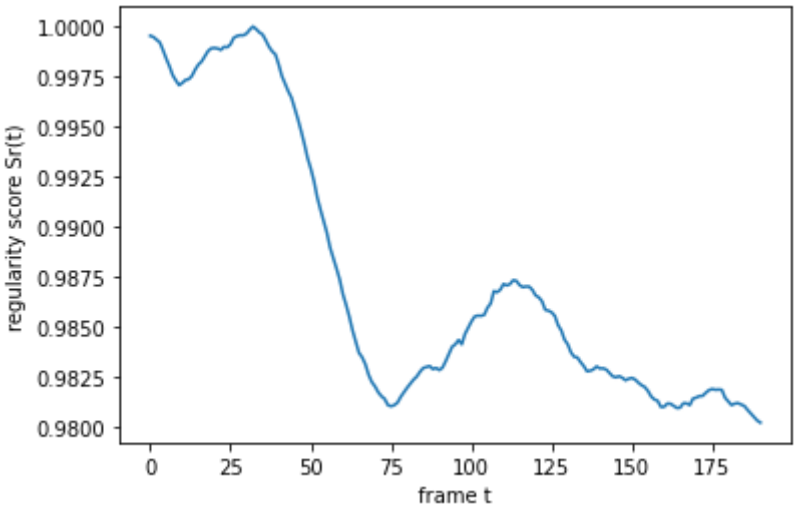



```
AUC:  0.8382675438596491
EER:  0.21875
EER THRESHOLD:  0.9907652767455326
Optimal threshold value is: 0.9910785402407789
```

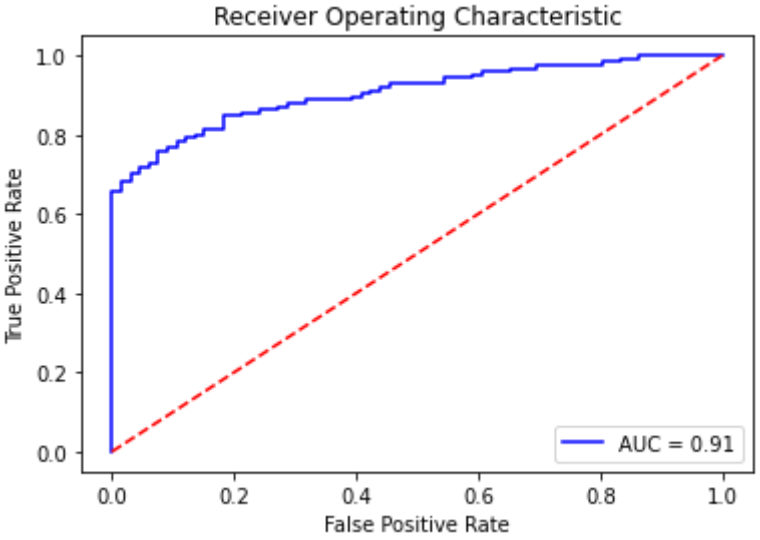


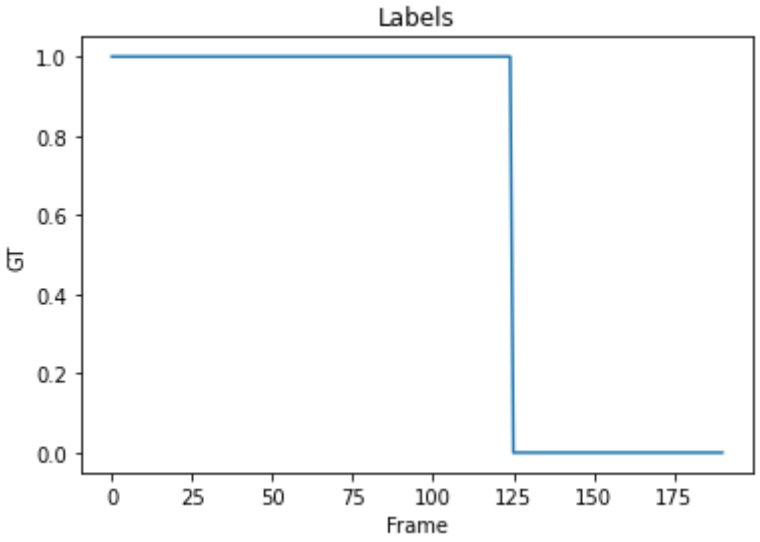



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test012
GT:  12
got model
(200, 227, 227, 1)
got data
```

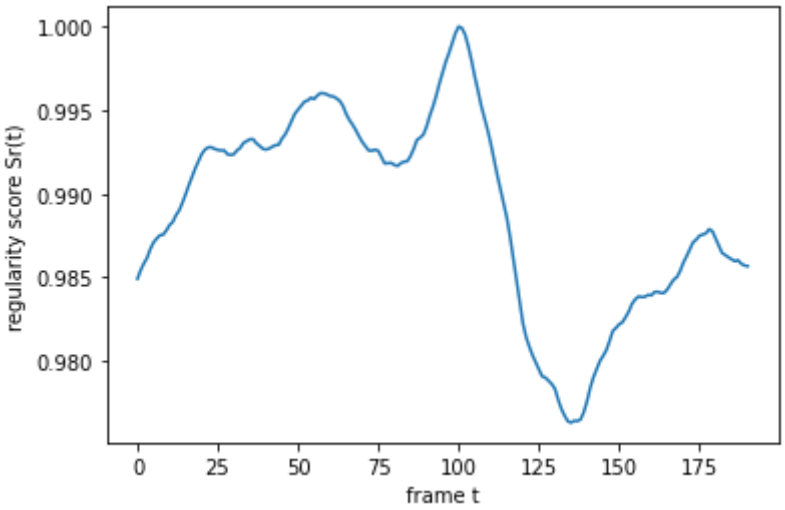



```
AUC:  0.9058181818181817
EER:  0.18181818181818182
EER THRESHOLD:  0.982991351485878
Optimal threshold value is: 0.984101674106553
```

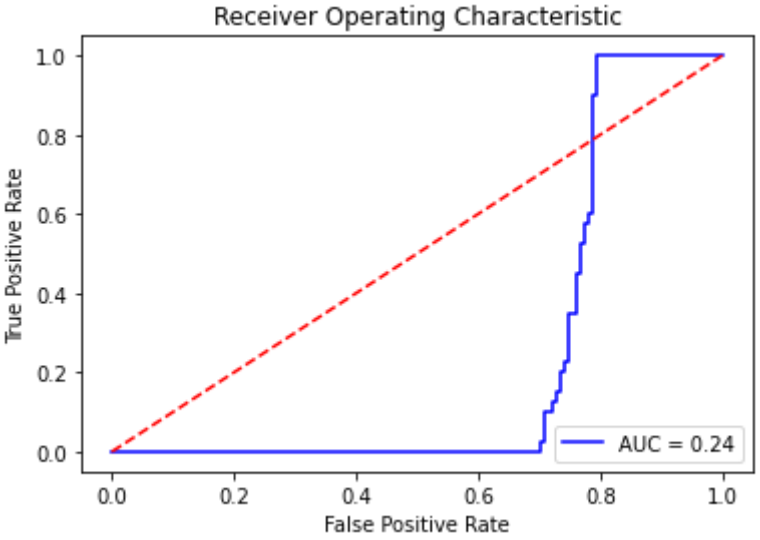


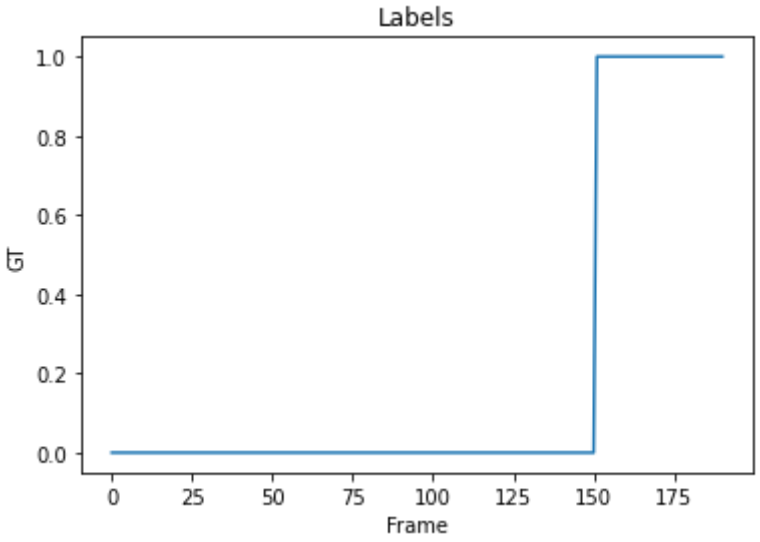



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test013
GT:  13
got model
(200, 227, 227, 1)
got data
```

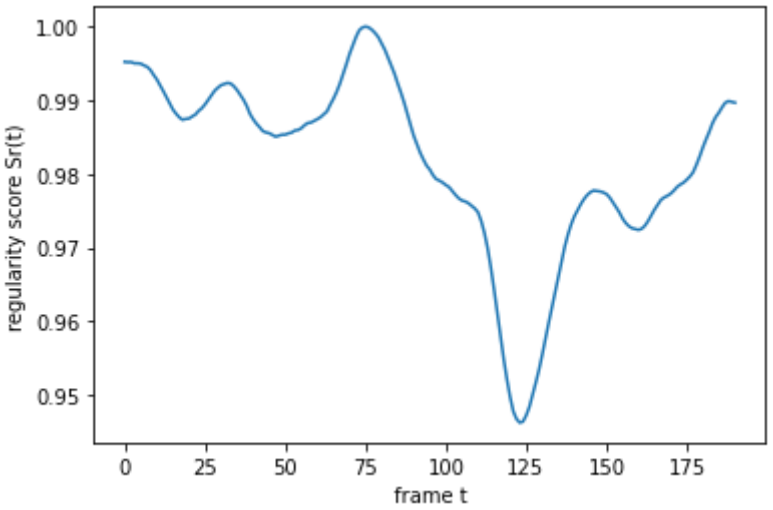



```
AUC:  0.23625827814569536
EER:  0.7483443708609272
EER THRESHOLD:  0.9866998018769568
Optimal threshold value is: 0.9823240747486538
```

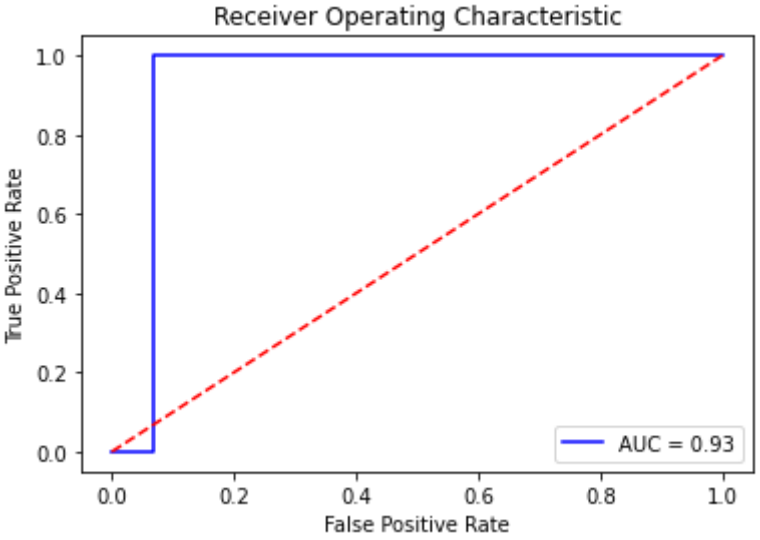


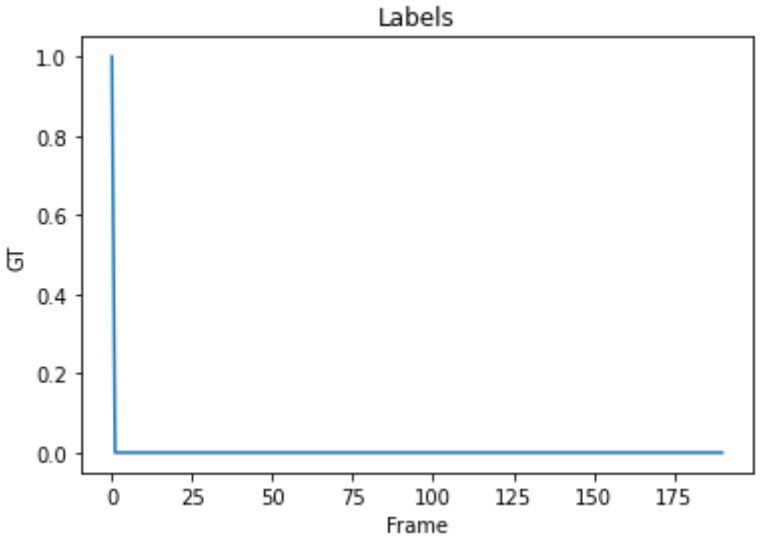



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test014
GT:  14
got model
(200, 227, 227, 1)
got data
```

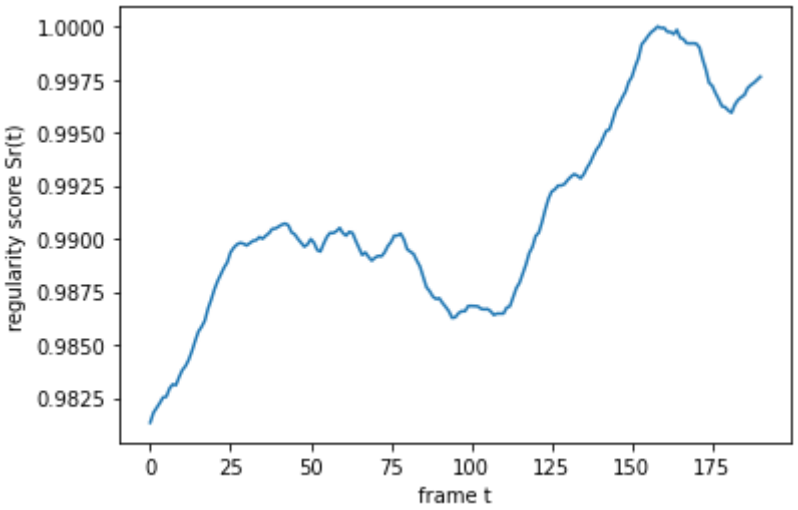



```
AUC:  0.9315789473684211
EER:  0.06842105263157895
EER THRESHOLD:  0.9952066444069388
Optimal threshold value is: 0.9952066444069388
```

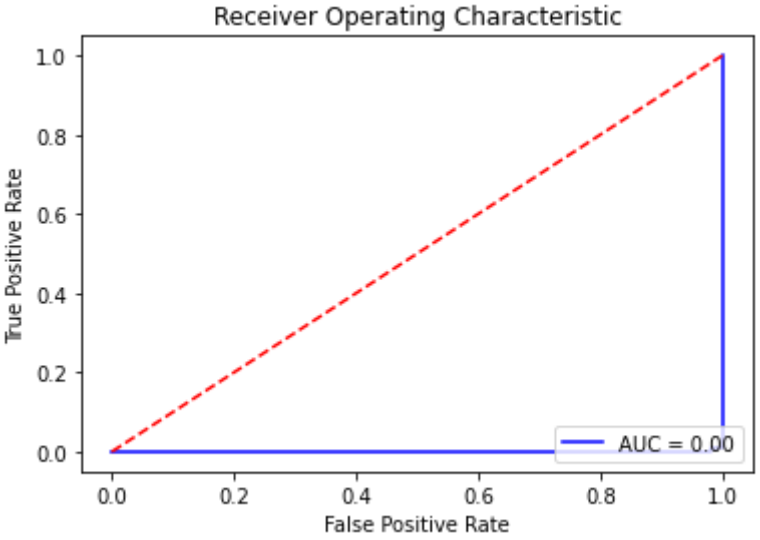


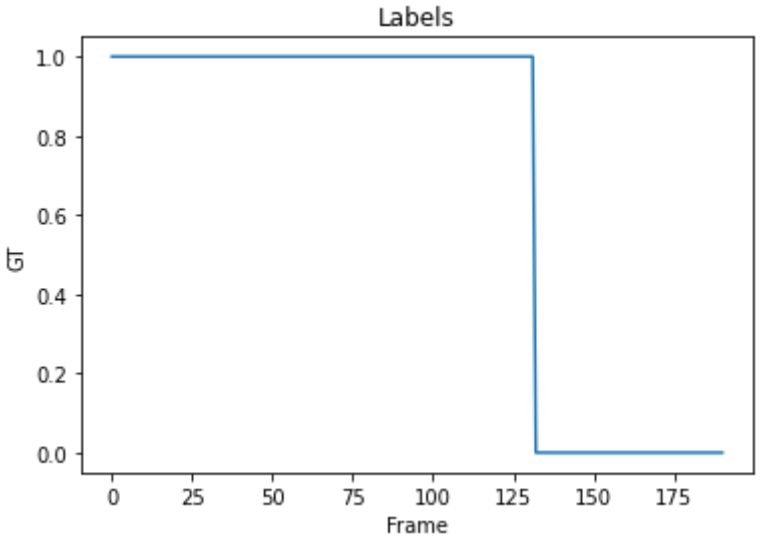



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test015
GT:  15
got model
(200, 227, 227, 1)
got data
```

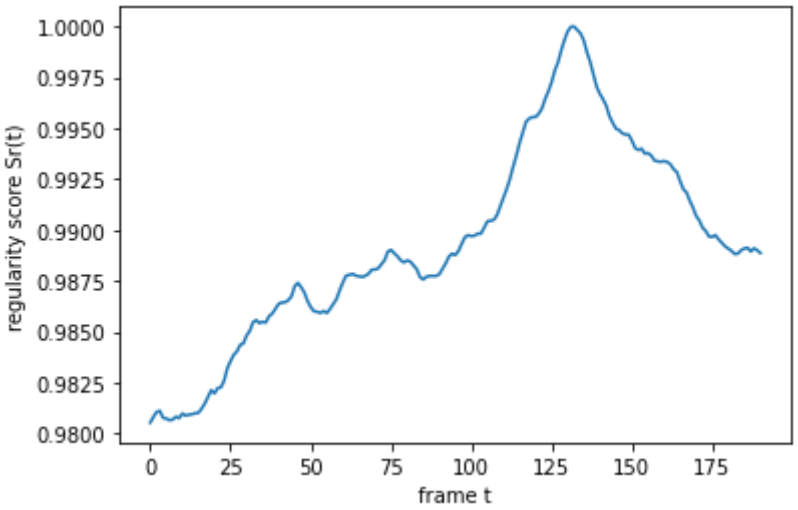



```
AUC:  0.00012840267077555178
EER:  1.0
EER THRESHOLD:  0.9928454465524243
Optimal threshold value is: 2.0
```

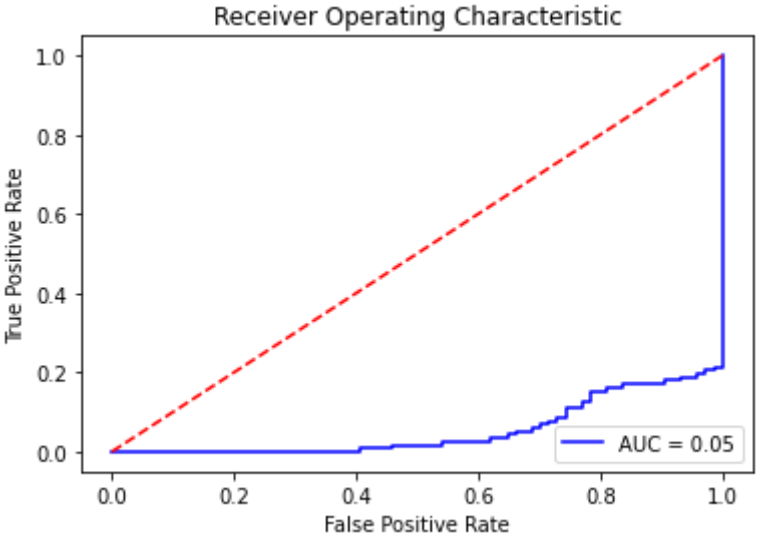


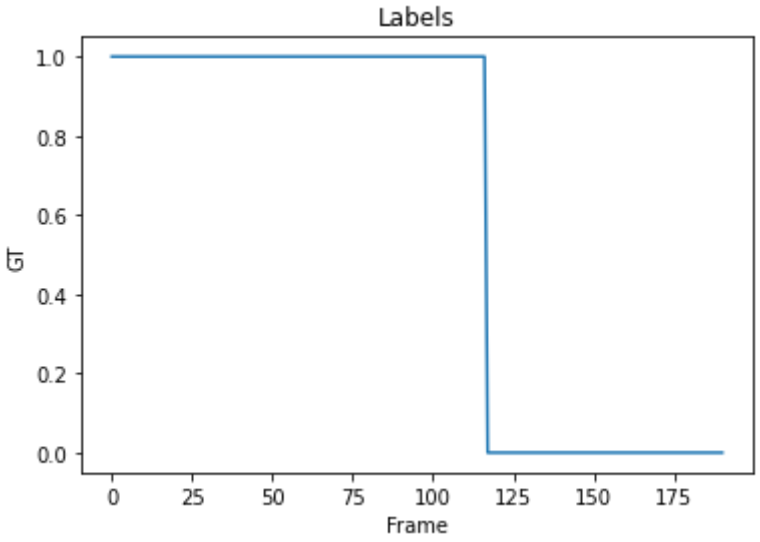



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test016
GT:  16
got model
(200, 227, 227, 1)
got data
```

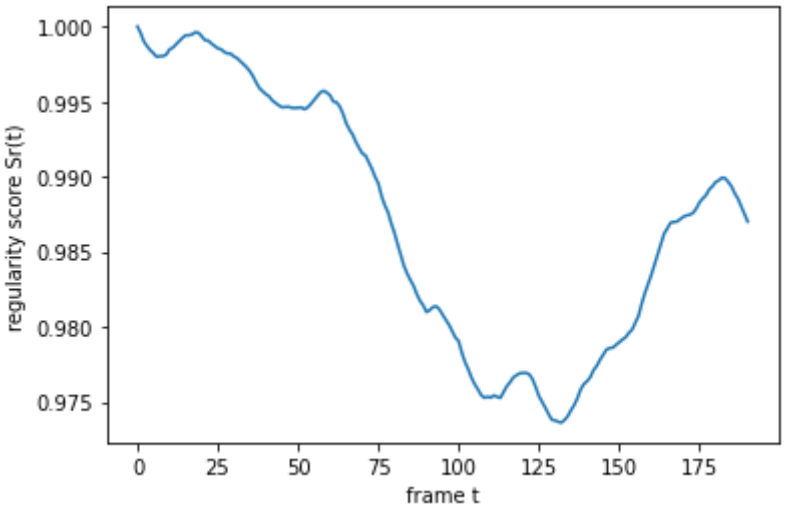



```
AUC:  0.05382305382305382
EER:  0.8378378378378378
EER THRESHOLD:  0.9893730536401902
Optimal threshold value is: 2.0
```

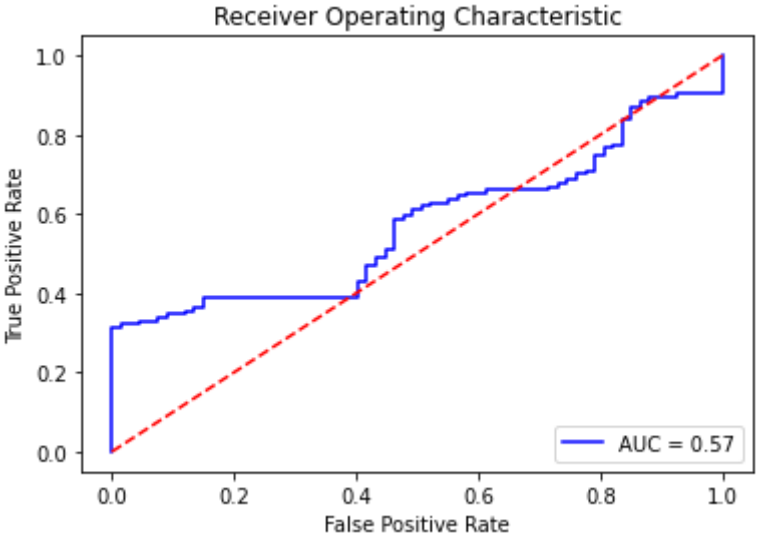


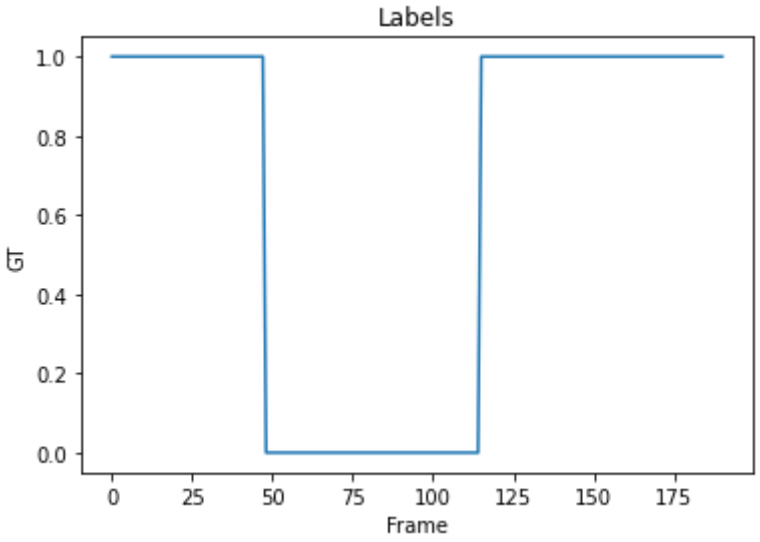



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test018
GT:  18
got model
(200, 227, 227, 1)
got data
```

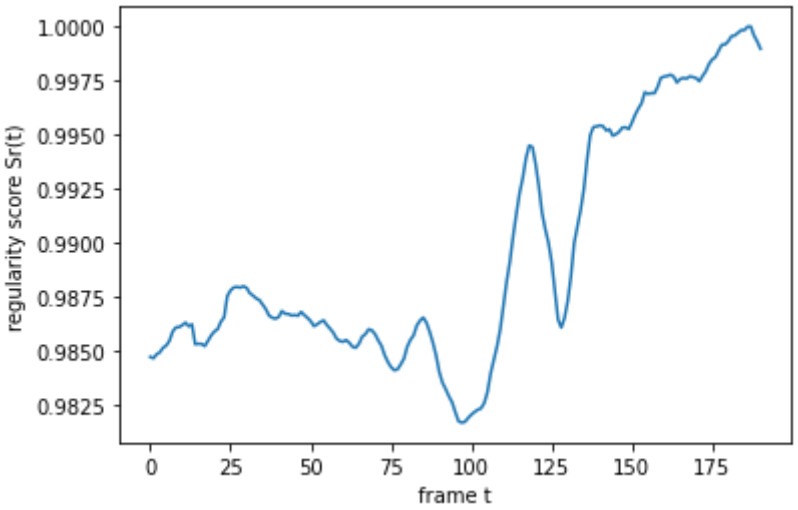



```
AUC:  0.5712566201251805
EER:  0.4626865671641791
EER THRESHOLD:  0.9876454458130696
Optimal threshold value is: 0.9958913902203497
```

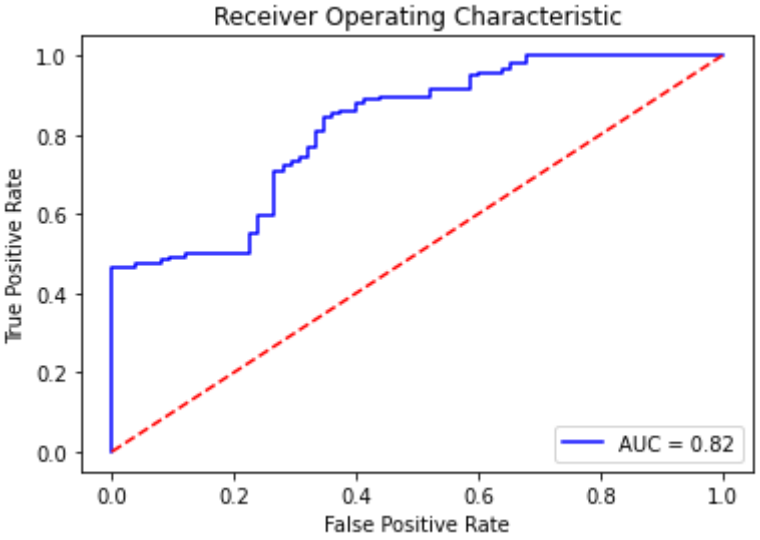


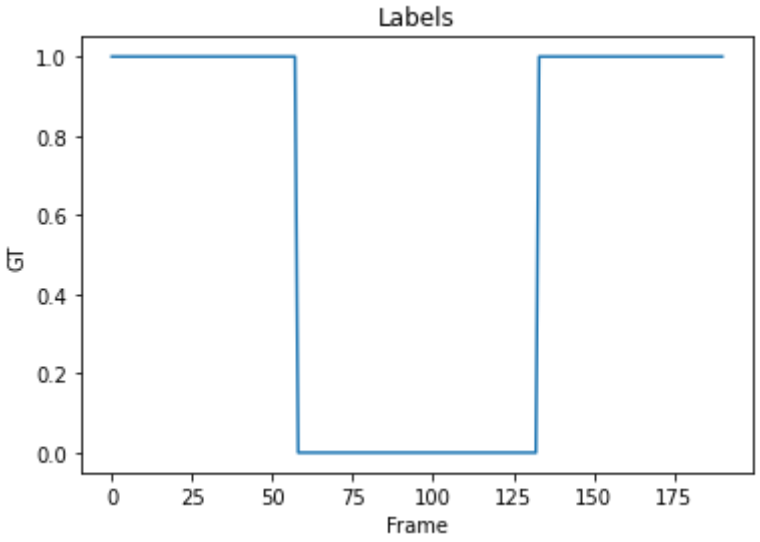



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test019
GT:  19
got model
(200, 227, 227, 1)
got data
```

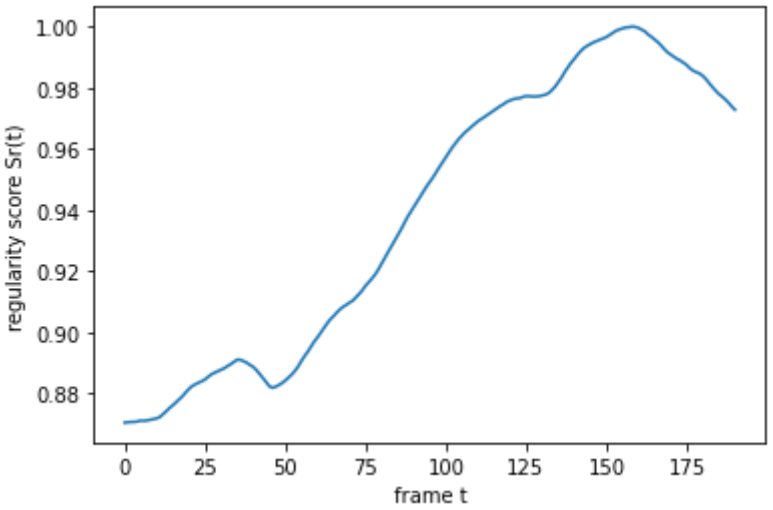



```
AUC:  0.8171264367816091
EER:  0.28
EER THRESHOLD:  0.986534978644878
Optimal threshold value is: 0.9860964694302833
```

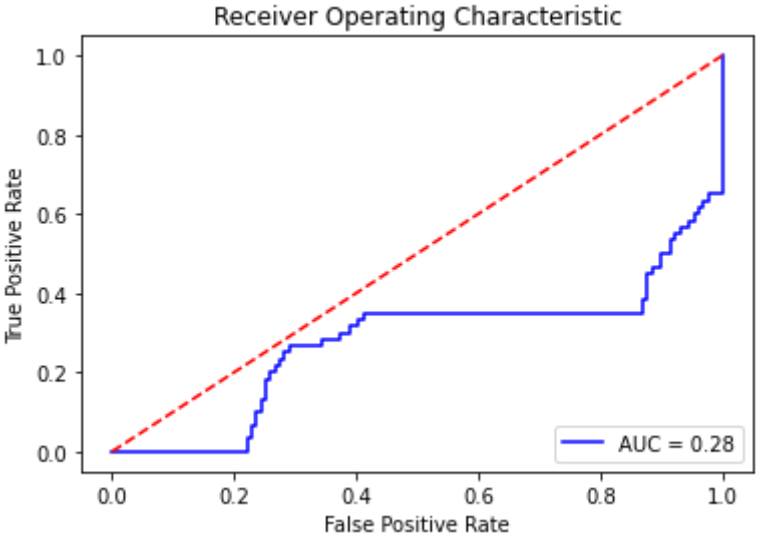


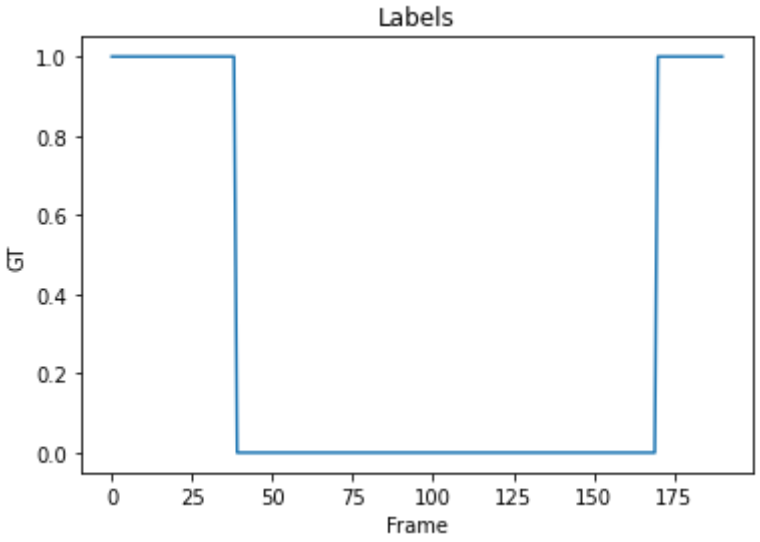



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test020
GT:  20
got model
(200, 227, 227, 1)
got data
```

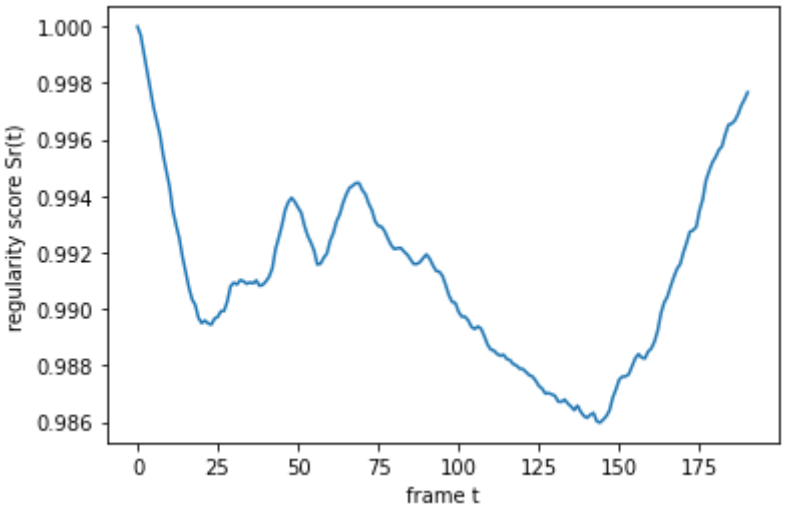



```
AUC:  0.2774809160305343
EER:  0.8702290076335878
EER THRESHOLD:  0.8922156837666864
Optimal threshold value is: 2.0
```

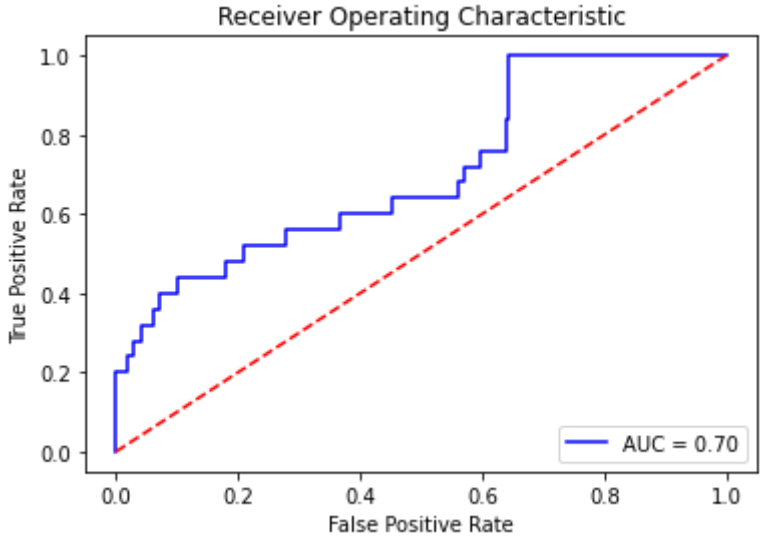


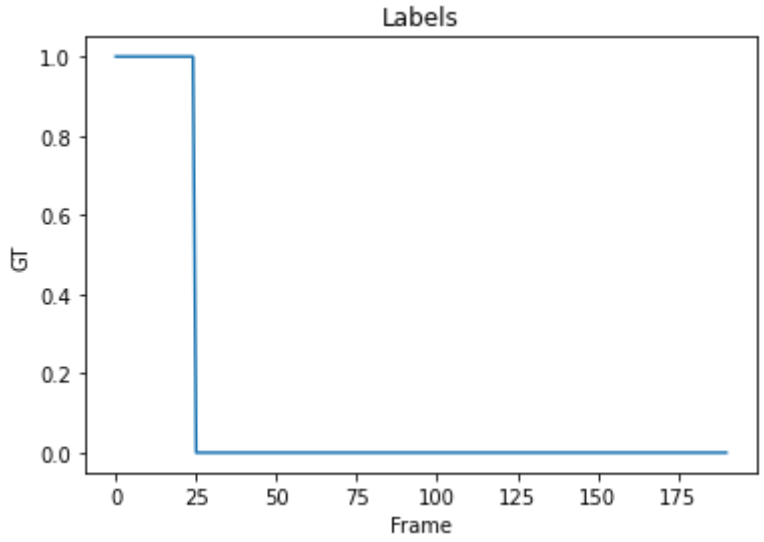



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test021
GT:  21
got model
(200, 227, 227, 1)
got data
```

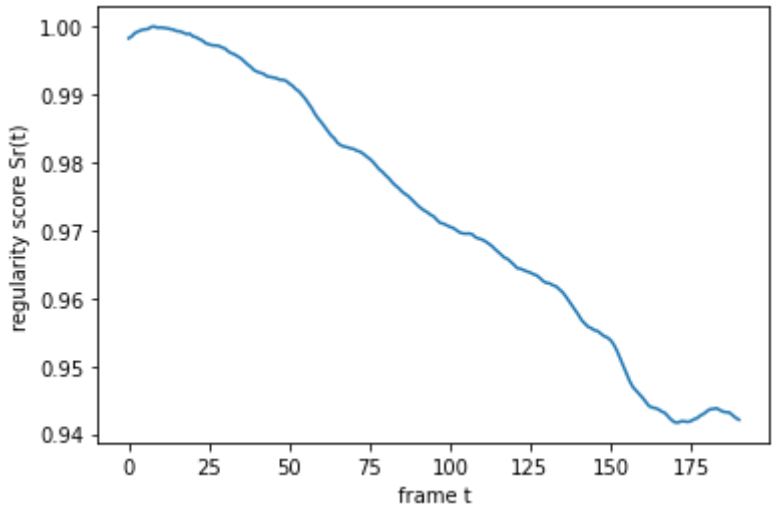



```
AUC:  0.7040963855421687
EER:  0.3674698795180723
EER THRESHOLD:  0.9918590272179445
Optimal threshold value is: 0.9894458403627324
```

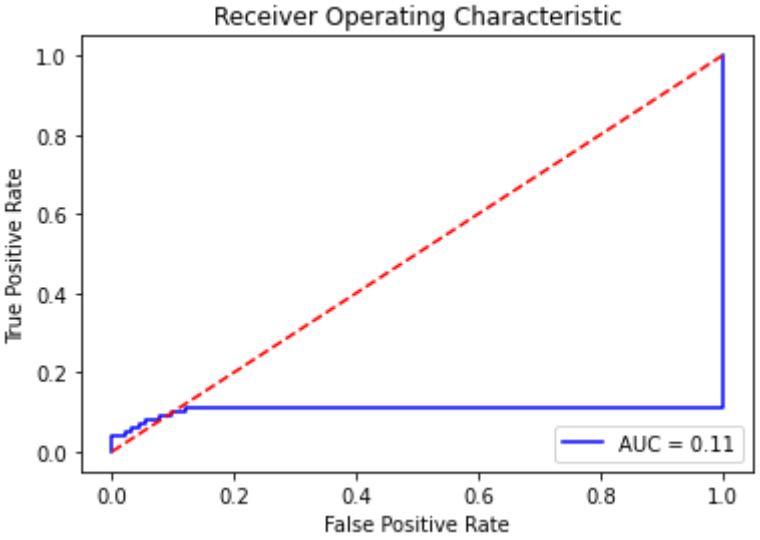


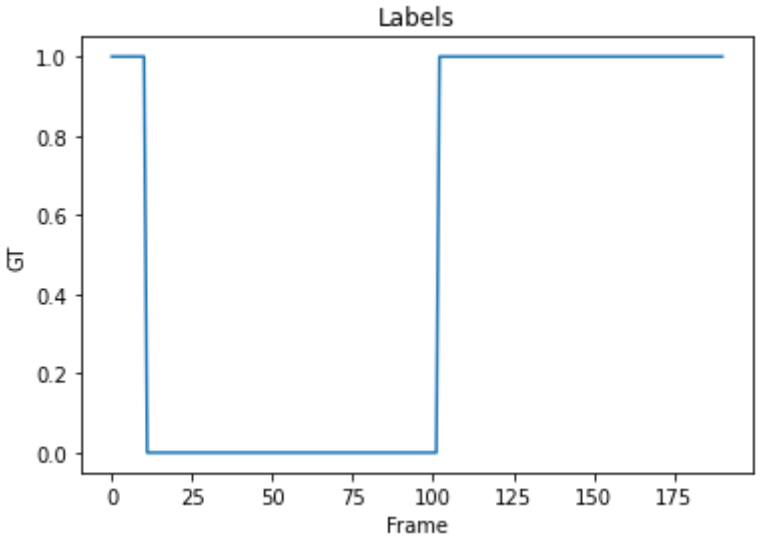



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test022
GT:  22
got model
(200, 227, 227, 1)
got data
```

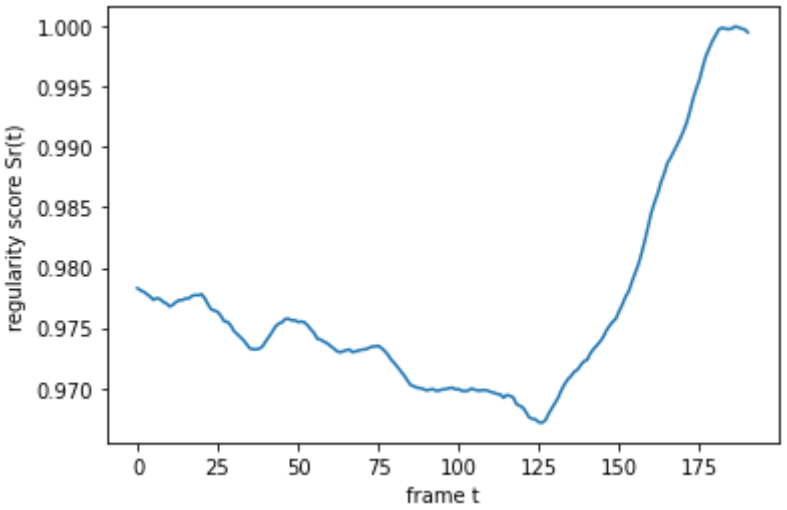



```
AUC:  0.1054945054945055
EER:  1.0
EER THRESHOLD:  0.970355586559704
Optimal threshold value is: 0.99982841985231
```

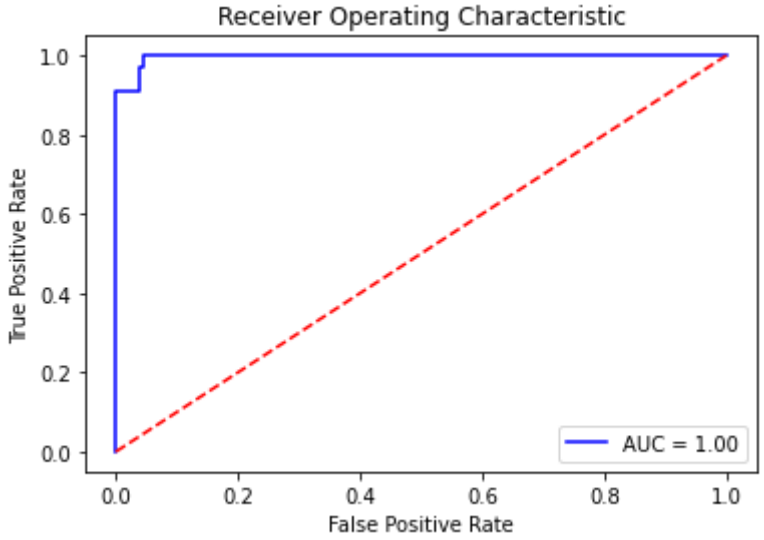


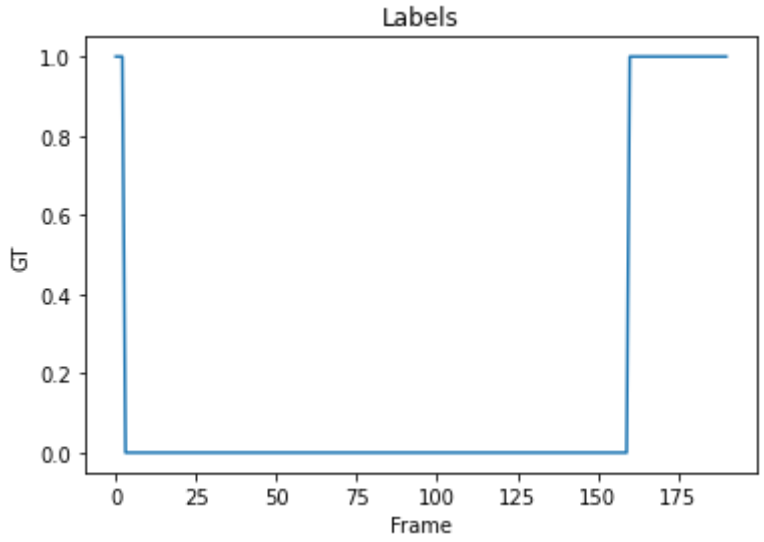



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test023
GT:  23
got model
(200, 227, 227, 1)
got data
```

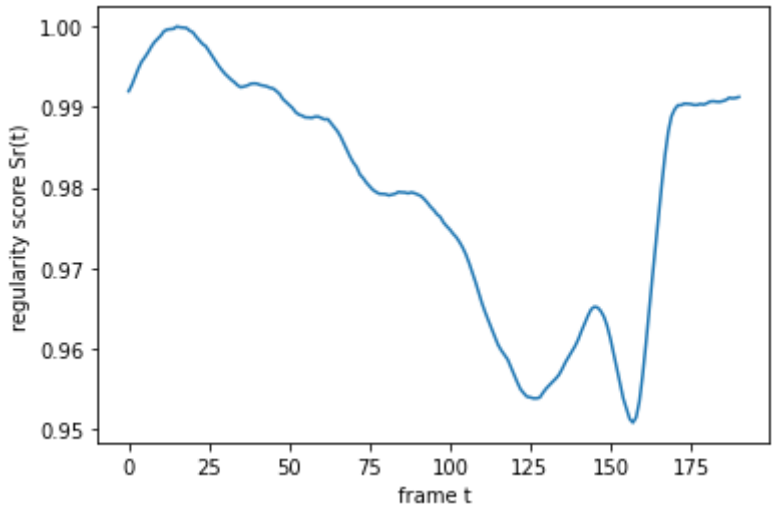



```
AUC:  0.9964406144623454
EER:  0.03821656050955414
EER THRESHOLD:  0.9781086808128847
Optimal threshold value is: 0.9779913085180866
```

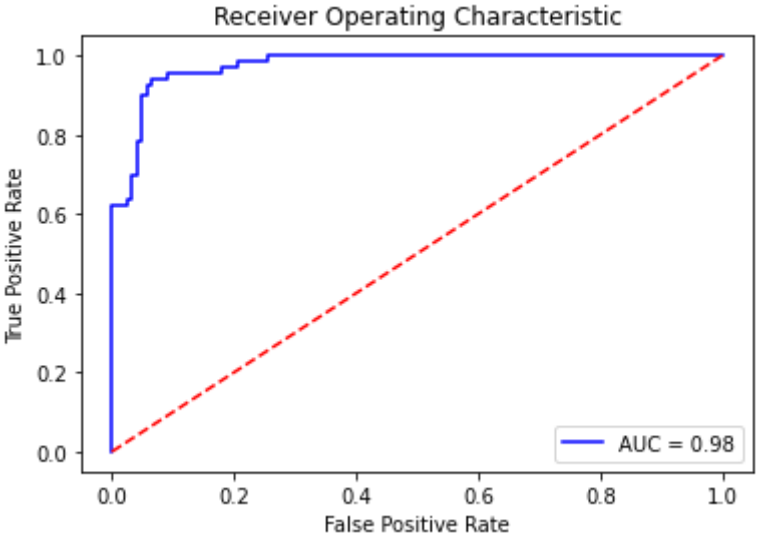


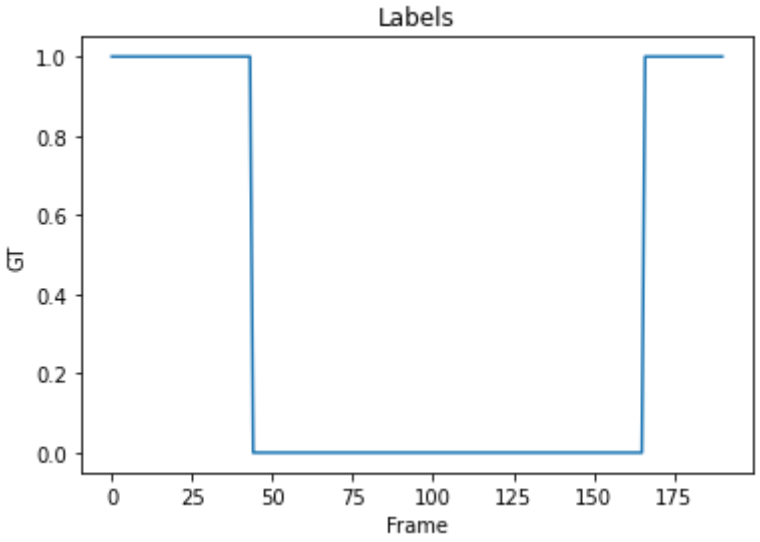



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test024
GT:  24
got model
(200, 227, 227, 1)
got data
```

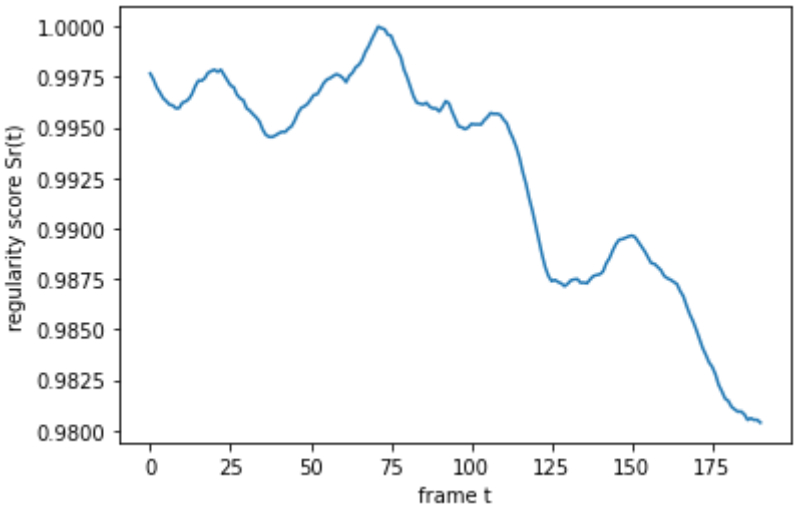



```
AUC:  0.9752910430030887
EER:  0.06557377049180328
EER THRESHOLD:  0.9898908844376024
Optimal threshold value is: 0.9897247620517174
```

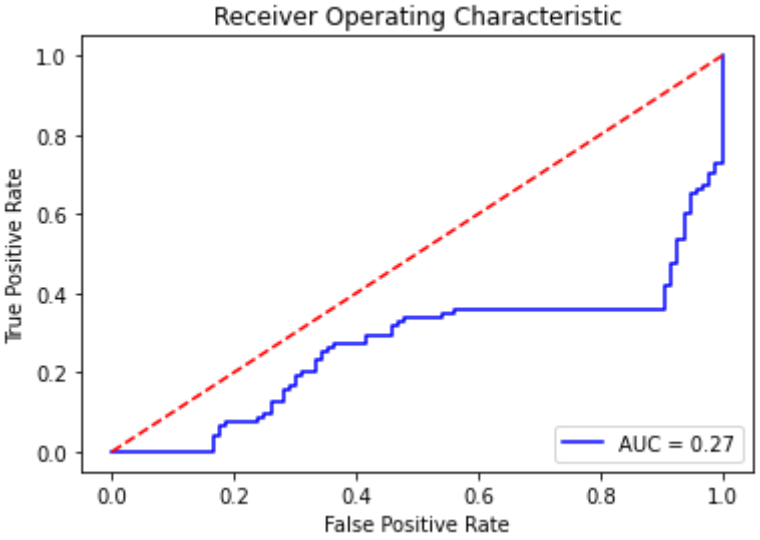


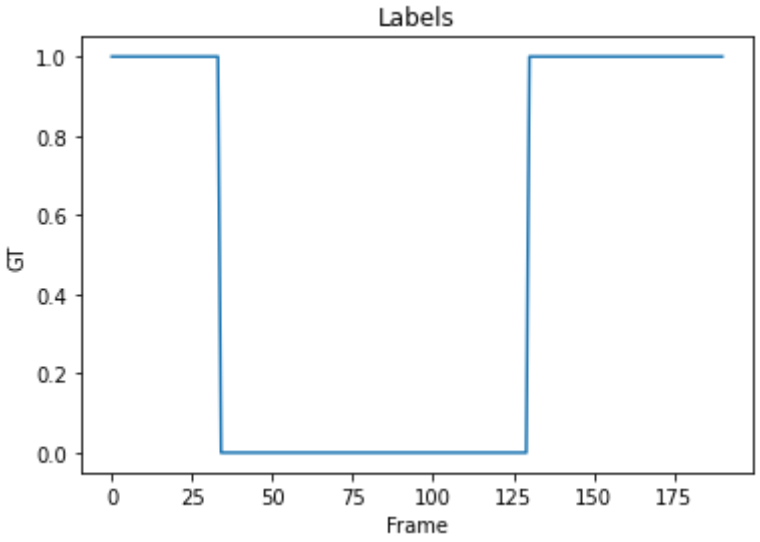



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test025
GT:  25
got model
(200, 227, 227, 1)
got data
```

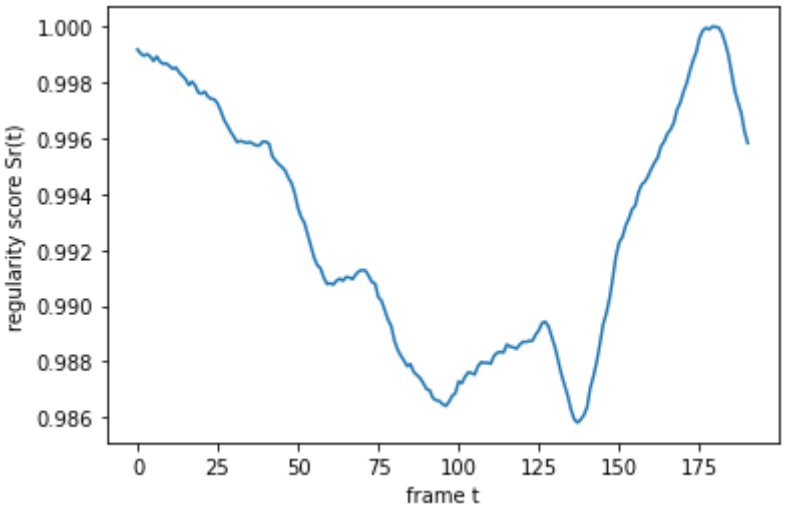



```
AUC:  0.2680921052631579
EER:  0.5625
EER THRESHOLD:  0.995503732924542
Optimal threshold value is: 2.0
```

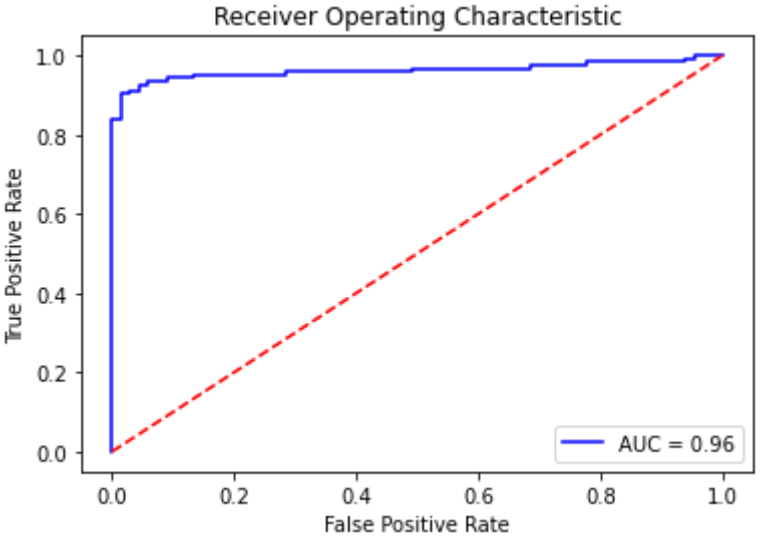


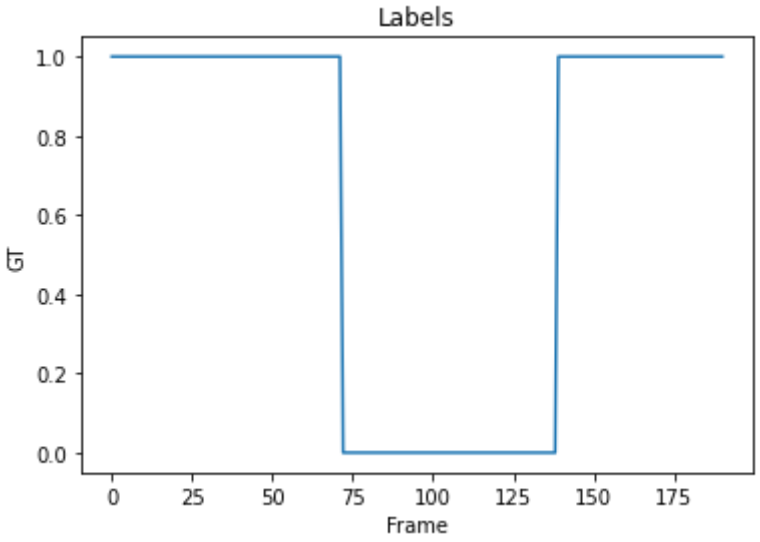



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test026
GT:  26
got model
(200, 227, 227, 1)
got data
```

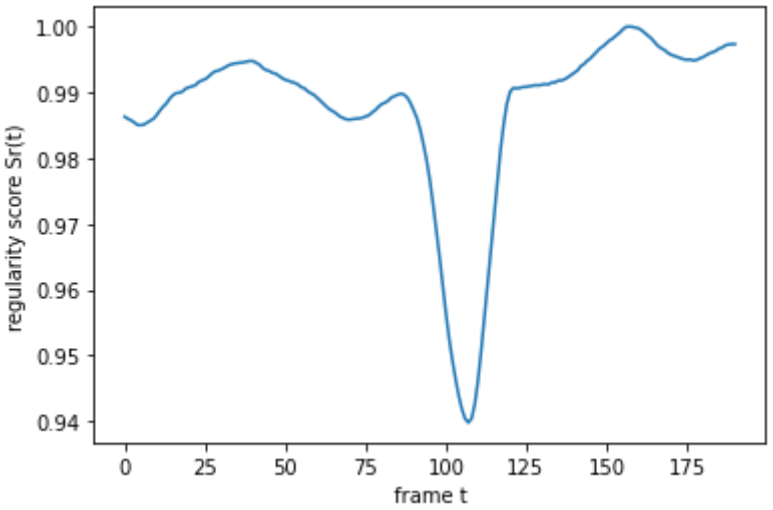



```
AUC:  0.9624458353394318
EER:  0.05970149253731343
EER THRESHOLD:  0.9902593591460529
Optimal threshold value is: 0.9908890980169331
```

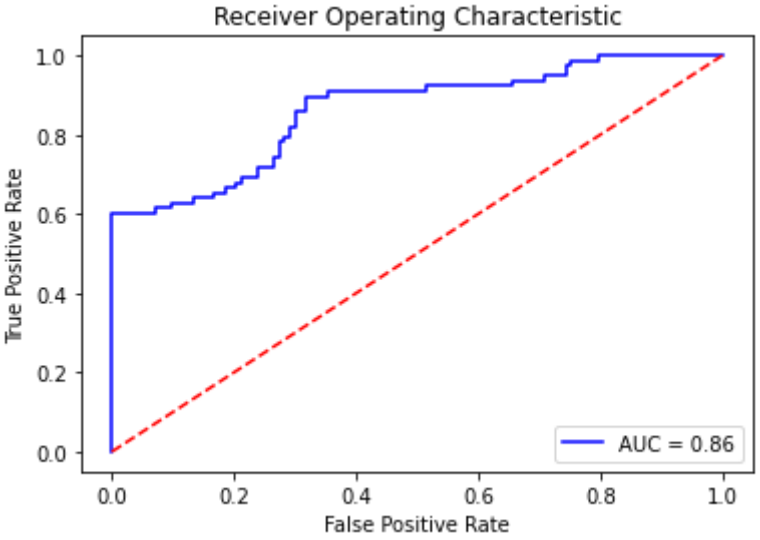


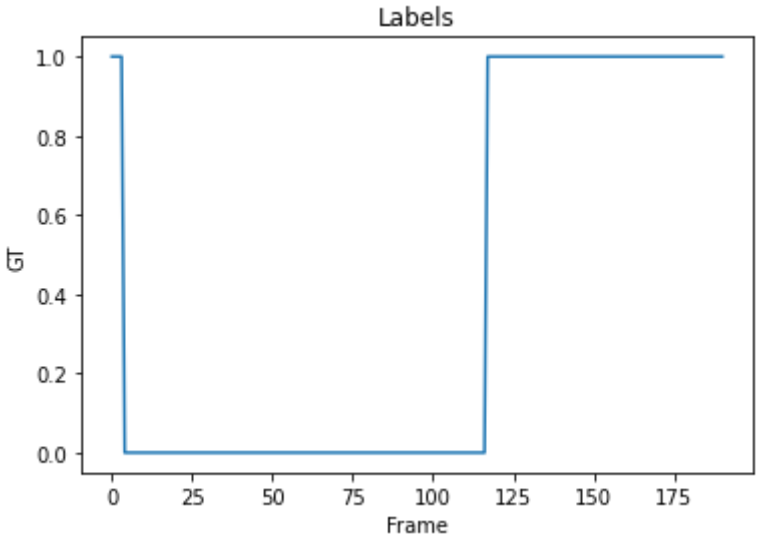



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test027
GT:  27
got model
(200, 227, 227, 1)
got data
```

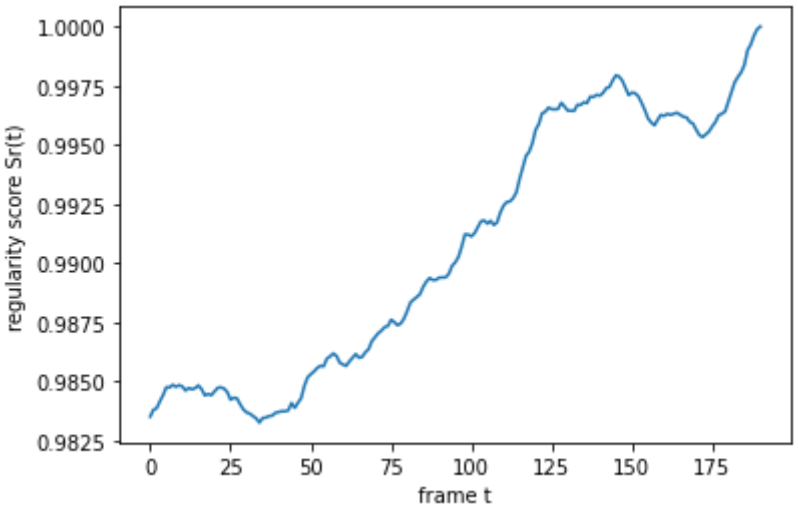



```
AUC:  0.8603358293623781
EER:  0.26548672566371684
EER THRESHOLD:  0.9914946407727718
Optimal threshold value is: 0.9947885465308167
```

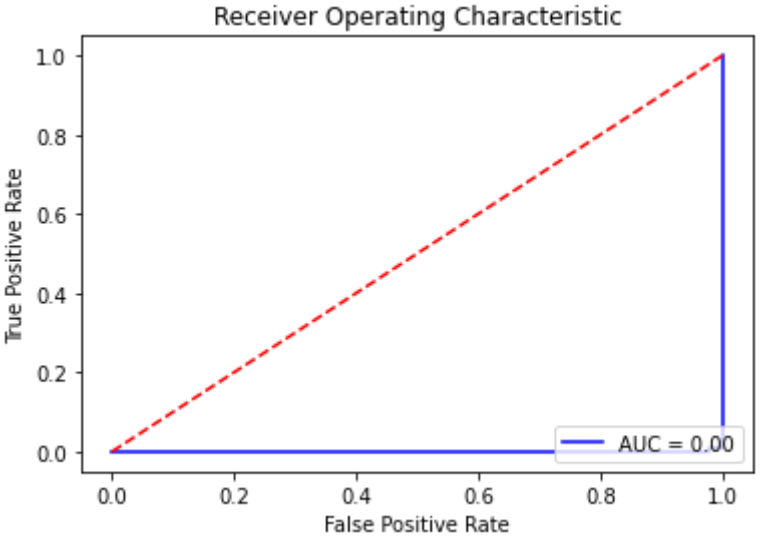


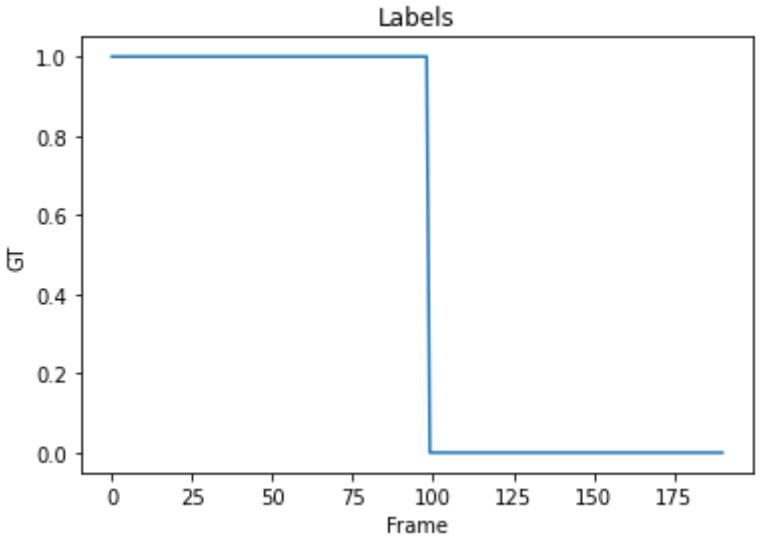



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test028
GT:  28
got model
(200, 227, 227, 1)
got data
```

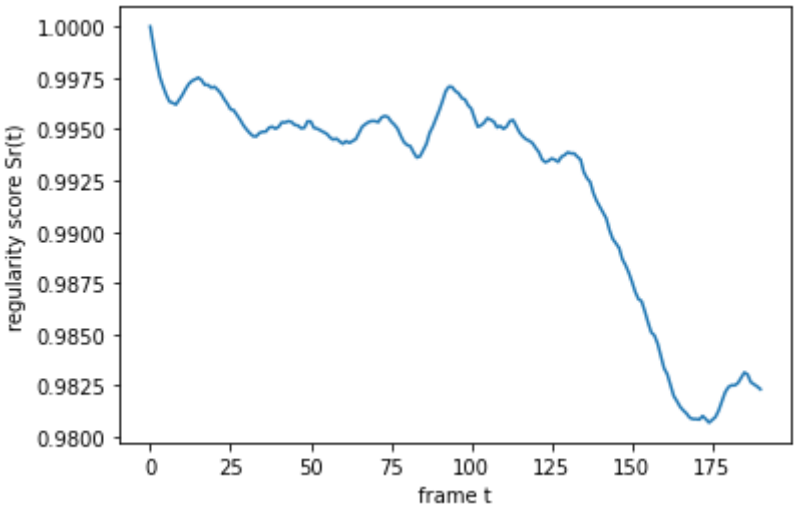



```
AUC:  0.0002195871761089151
EER:  1.0
EER THRESHOLD:  0.9911273550390515
Optimal threshold value is: 2.0
```

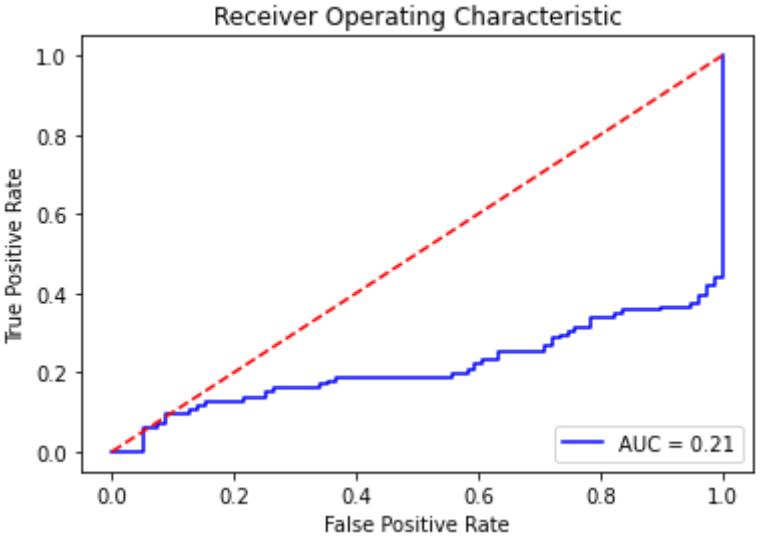


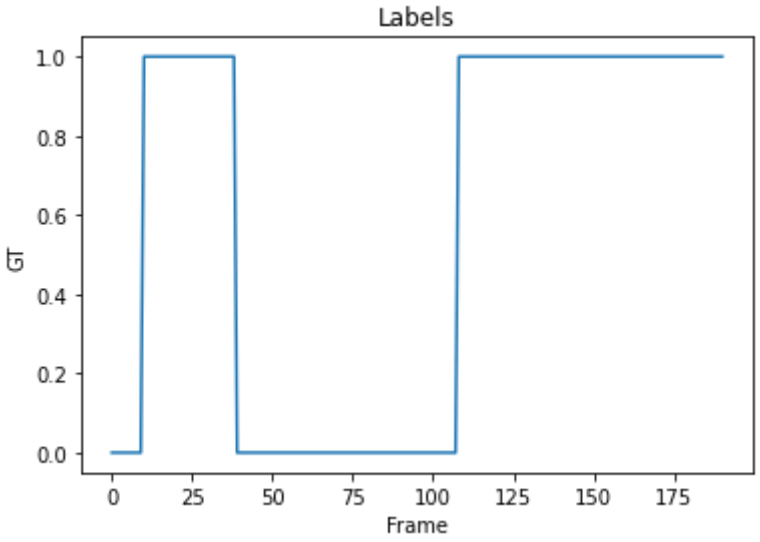



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test029
GT:  29
got model
(200, 227, 227, 1)
got data
```

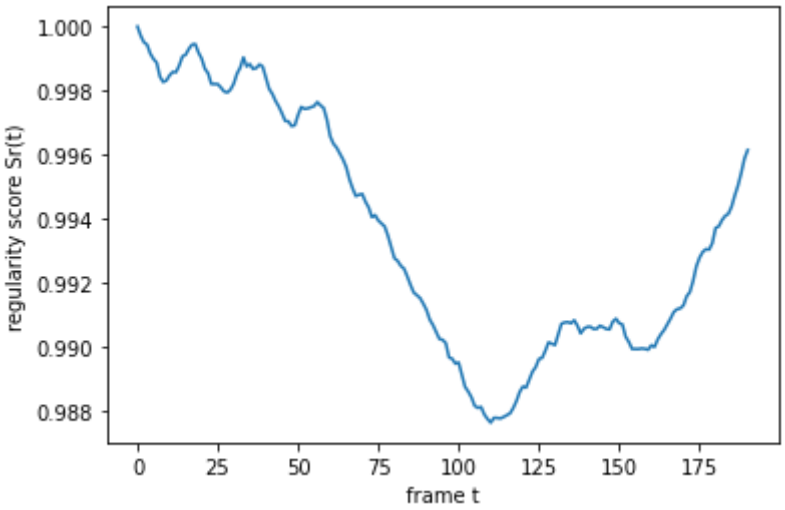



```
AUC:  0.21292947558770342
EER:  0.7215189873417721
EER THRESHOLD:  0.9948742595815183
Optimal threshold value is: 0.9971558056550506
```

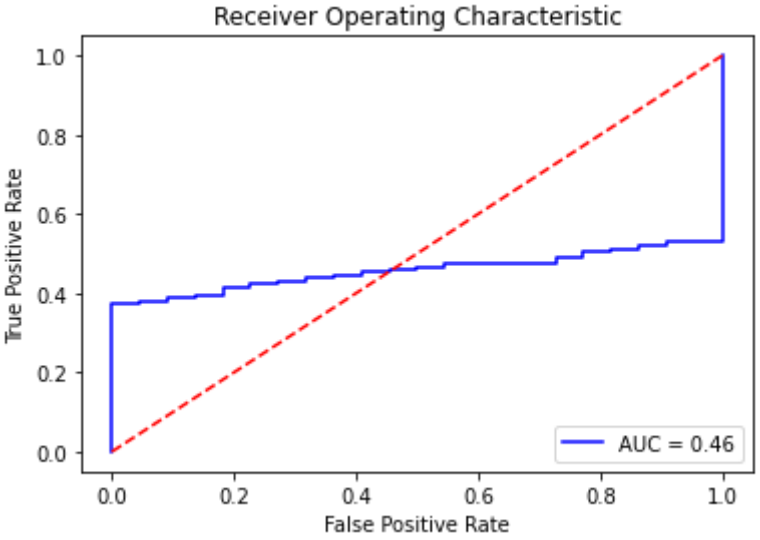


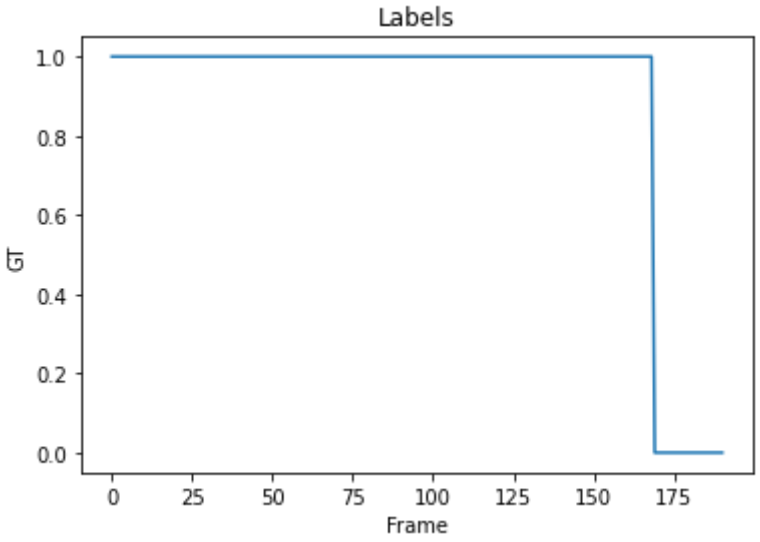



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test030
GT:  30
got model
(200, 227, 227, 1)
got data
```

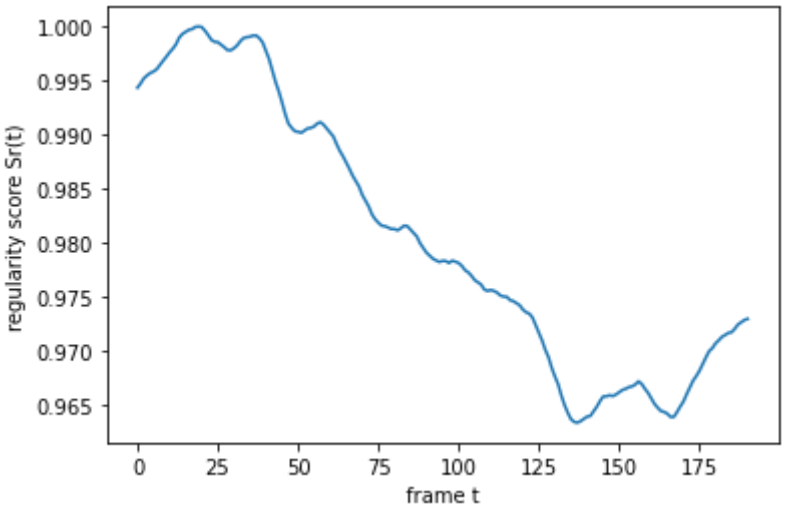



```
AUC:  0.457235072619688
EER:  0.5454545454545454
EER THRESHOLD:  0.9932350835350059
Optimal threshold value is: 0.9962206515663683
```

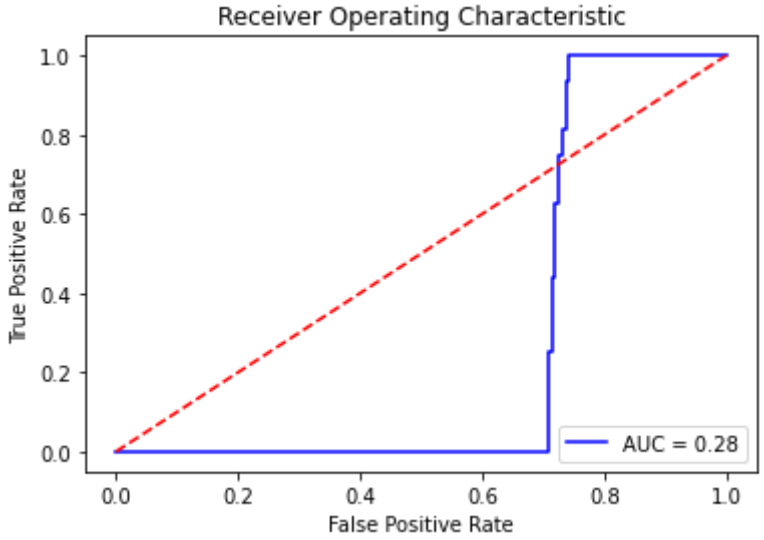


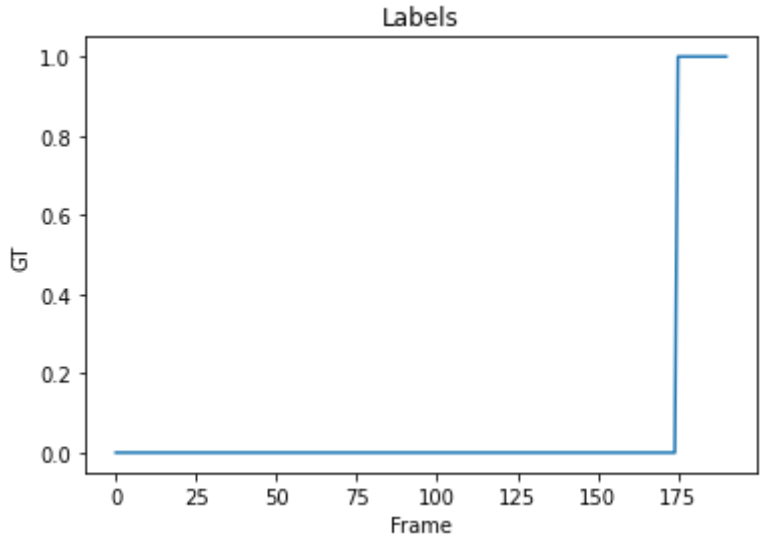



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test031
GT:  31
got model
(200, 227, 227, 1)
got data
```

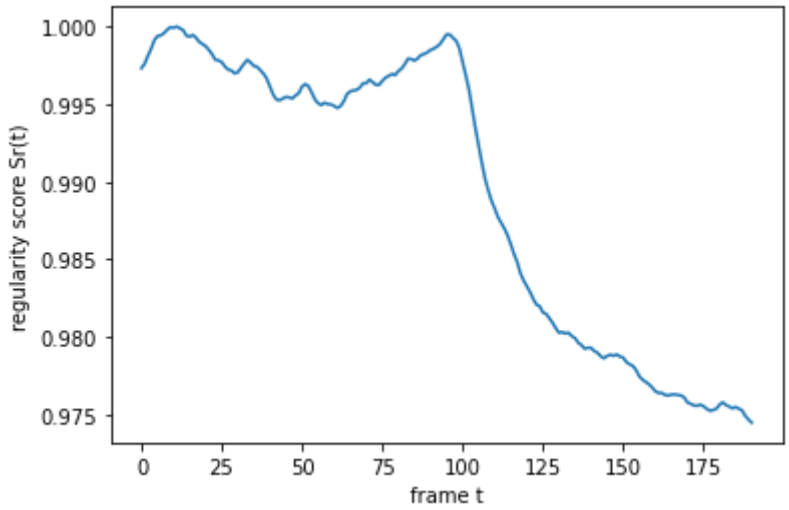



```
AUC:  0.2789285714285714
EER:  0.7142857142857143
EER THRESHOLD:  0.9723744536171058
Optimal threshold value is: 0.9681724302096228
```

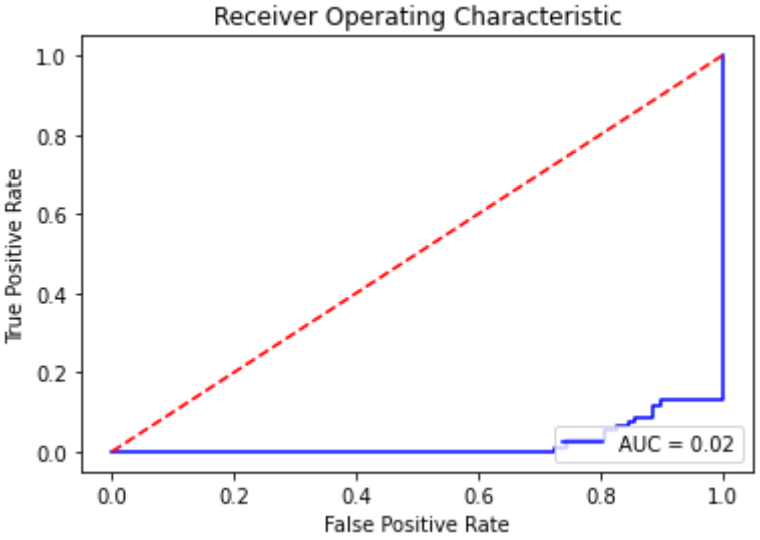


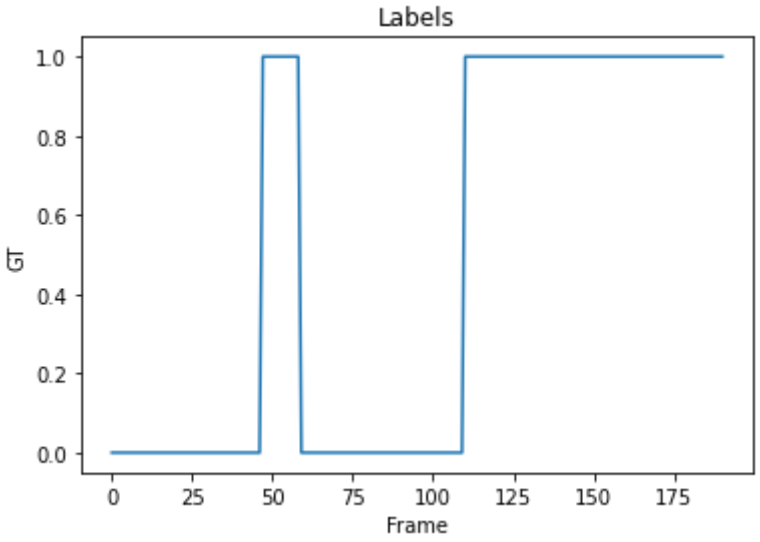



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test032
GT:  32
got model
(200, 227, 227, 1)
got data
```

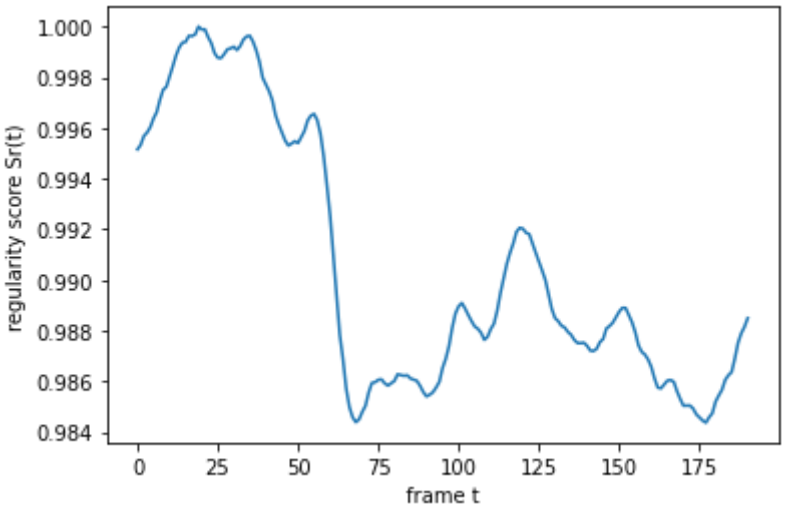



```
AUC:  0.02238314680710994
EER:  0.8877551020408163
EER THRESHOLD:  0.9950030889292698
Optimal threshold value is: 2.0
```

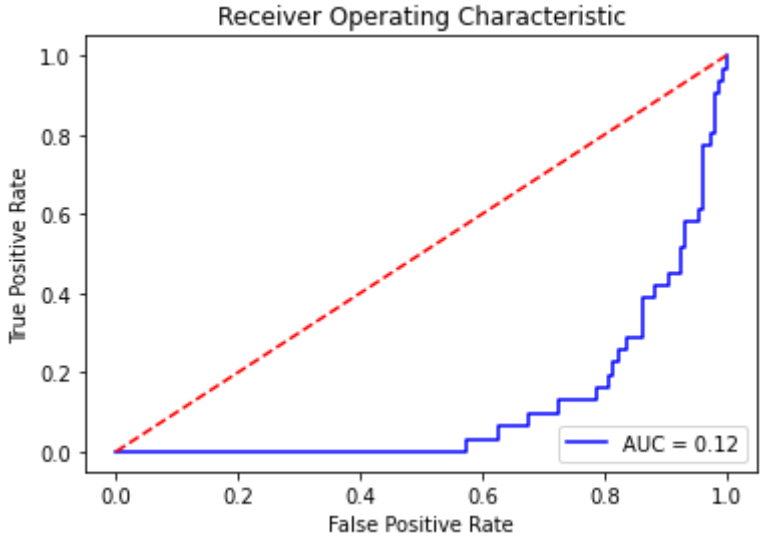


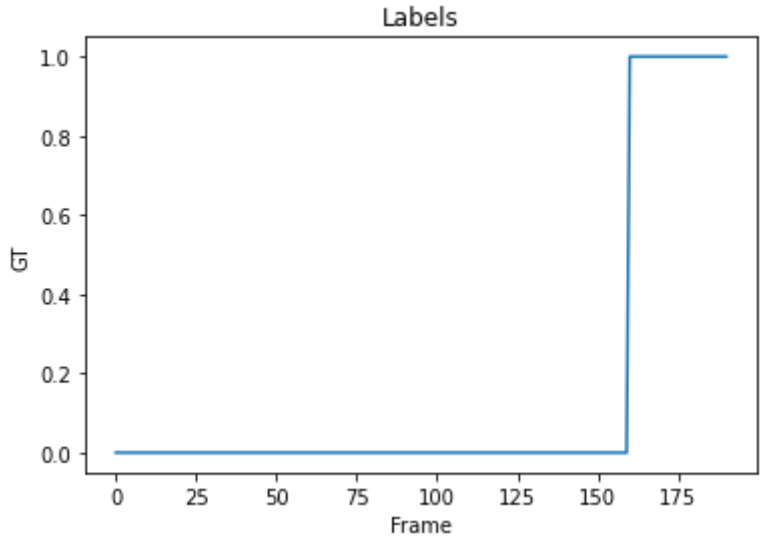



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test033
GT:  33
got model
(200, 227, 227, 1)
got data
```

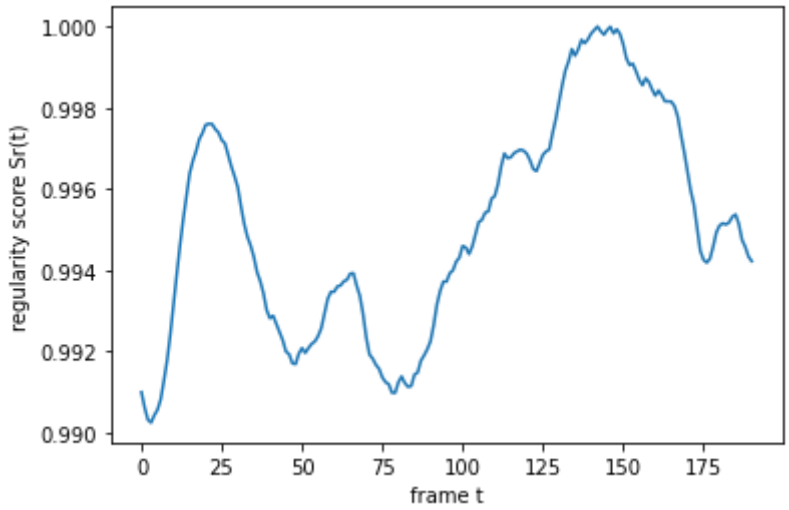



```
AUC:  0.11532258064516128
EER:  0.80625
EER THRESHOLD:  0.9865664256914763
Optimal threshold value is: 2.0
```

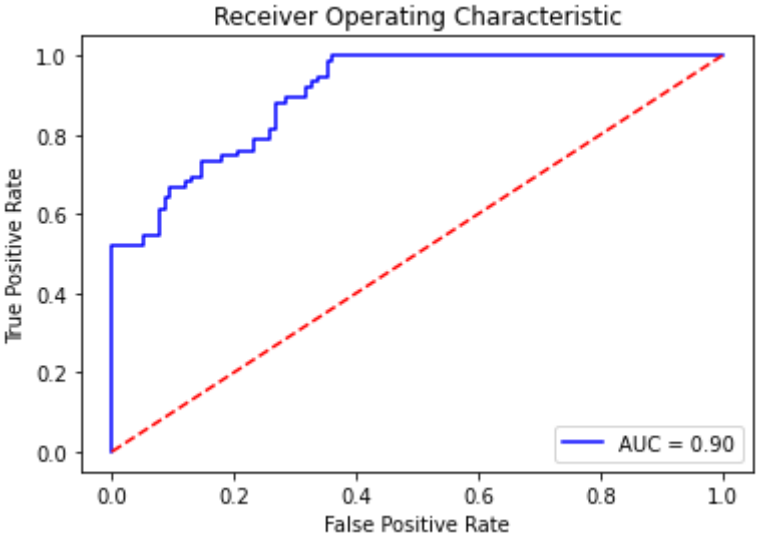


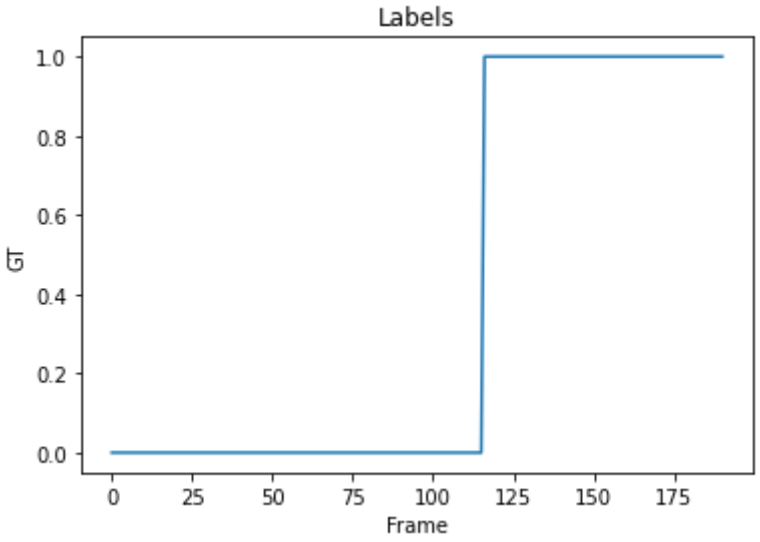



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test034
GT:  34
got model
(200, 227, 227, 1)
got data
```

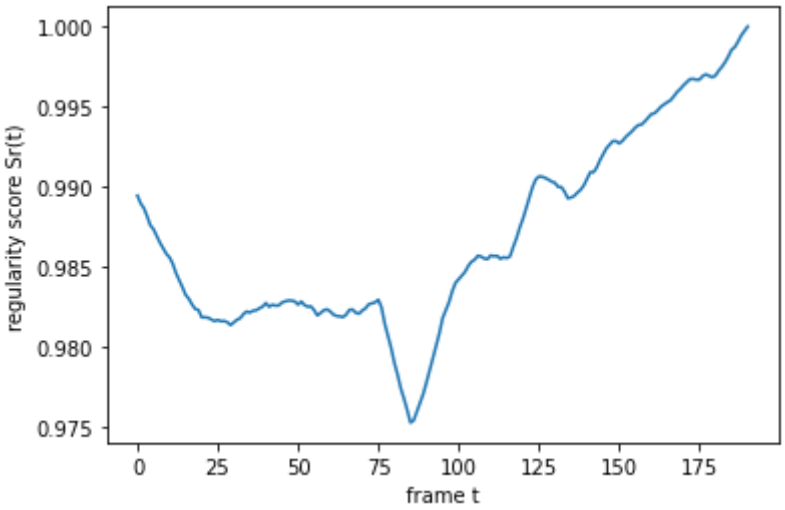



```
AUC:  0.9032183908045978
EER:  0.23275862068965517
EER THRESHOLD:  0.9954172873910175
Optimal threshold value is: 0.994187923743158
```

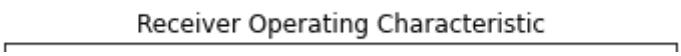





```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test035
GT:  35
got model
(200, 227, 227, 1)
got data
```




```
AUC:  0.17804054054054053
EER:  0.8288288288288288
EER THRESHOLD:  0.9839758402260982
Optimal threshold value is: 0.9799073635766546
```



In [ ]: