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

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_v9.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, padding
```

```
    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 16:14:30.379940: 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 16:14:30.379991: 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(s
    sa = (sequences_reconstruction_cost - np.min(sequences_reconstruction_
    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]:

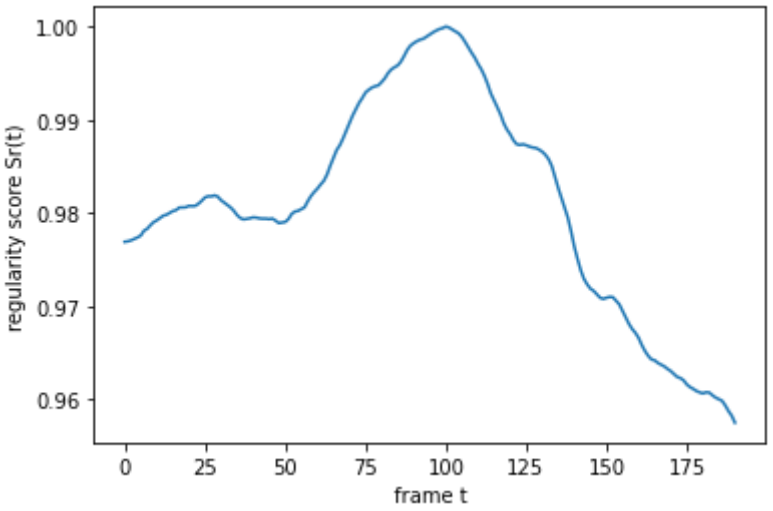
```
pr, before_reconstuction = evaluate(reload_model=True)
```

```
2021-11-11 16:14:58.157322: 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 16:14:58.157383: W tensorflow/stream_executor/cuda/cuda_driver.c
c:269] failed call to cuInit: UNKNOWN ERROR (303)
2021-11-11 16:14:58.157413: 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 16:14:58.157804: 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)

 time_distributed_4 (TimeDis  (None, 10, 227, 227, 1)  122
 tributed)

=================================================================
Total params: 1,069,309
Trainable params: 1,069,309
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 [==============================] - 734s 32s/step - loss: 0.0550
Epoch 2/50
22/22 [==============================] - 706s 32s/step - loss: 0.0472
Epoch 3/50
22/22 [==============================] - 692s 31s/step - loss: 0.0415
Epoch 4/50
22/22 [==============================] - 698s 32s/step - loss: 0.0412
Epoch 5/50
22/22 [==============================] - 708s 32s/step - loss: 0.0411
Epoch 6/50
22/22 [==============================] - 703s 32s/step - loss: 0.0411
Epoch 7/50
22/22 [==============================] - 698s 32s/step - loss: 0.0410
Epoch 8/50
22/22 [==============================] - 715s 32s/step - loss: 0.0410
Epoch 9/50
22/22 [==============================] - 709s 32s/step - loss: 0.0410
Epoch 10/50
```

```
22/22 [==============================] - 704s 32s/step - loss: 0.0409
Epoch 11/50
22/22 [==============================] - 717s 33s/step - loss: 0.0409
Epoch 12/50
22/22 [==============================] - 703s 32s/step - loss: 0.0409
Epoch 13/50
22/22 [==============================] - 708s 32s/step - loss: 0.0408
Epoch 14/50
22/22 [==============================] - 713s 32s/step - loss: 0.0408
Epoch 15/50
22/22 [==============================] - 708s 32s/step - loss: 0.0407
Epoch 16/50
22/22 [==============================] - 713s 32s/step - loss: 0.0406
Epoch 17/50
22/22 [==============================] - 714s 32s/step - loss: 0.0406
Epoch 18/50
22/22 [==============================] - 705s 32s/step - loss: 0.0405
Epoch 19/50
22/22 [==============================] - 709s 32s/step - loss: 0.0405
Epoch 20/50
22/22 [==============================] - 712s 32s/step - loss: 0.0404
Epoch 21/50
22/22 [==============================] - 704s 32s/step - loss: 0.0403
Epoch 22/50
22/22 [==============================] - 705s 32s/step - loss: 0.0402
Epoch 23/50
22/22 [==============================] - 706s 32s/step - loss: 0.0401
Epoch 24/50
22/22 [==============================] - 699s 32s/step - loss: 0.0400
Epoch 25/50
22/22 [==============================] - 710s 32s/step - loss: 0.0399
Epoch 26/50
22/22 [==============================] - 715s 33s/step - loss: 0.0398
Epoch 27/50
22/22 [==============================] - 700s 32s/step - loss: 0.0397
Epoch 28/50
22/22 [==============================] - 707s 32s/step - loss: 0.0396
Epoch 29/50
22/22 [==============================] - 711s 32s/step - loss: 0.0395
Epoch 30/50
22/22 [==============================] - 702s 32s/step - loss: 0.0394
Epoch 31/50
22/22 [==============================] - 710s 32s/step - loss: 0.0393
Epoch 32/50
22/22 [==============================] - 710s 32s/step - loss: 0.0392
Epoch 33/50
22/22 [==============================] - 700s 32s/step - loss: 0.0392
Epoch 34/50
22/22 [==============================] - 711s 32s/step - loss: 0.0391
Epoch 35/50
22/22 [==============================] - 705s 32s/step - loss: 0.0390
Epoch 36/50
22/22 [==============================] - 700s 32s/step - loss: 0.0390
Epoch 37/50
22/22 [==============================] - 714s 32s/step - loss: 0.0389
Epoch 38/50
22/22 [==============================] - 705s 32s/step - loss: 0.0389
Epoch 39/50
22/22 [==============================] - 697s 32s/step - loss: 0.0389
Epoch 40/50
22/22 [==============================] - 711s 32s/step - loss: 0.0388
Epoch 41/50
22/22 [==============================] - 705s 32s/step - loss: 0.0388
```

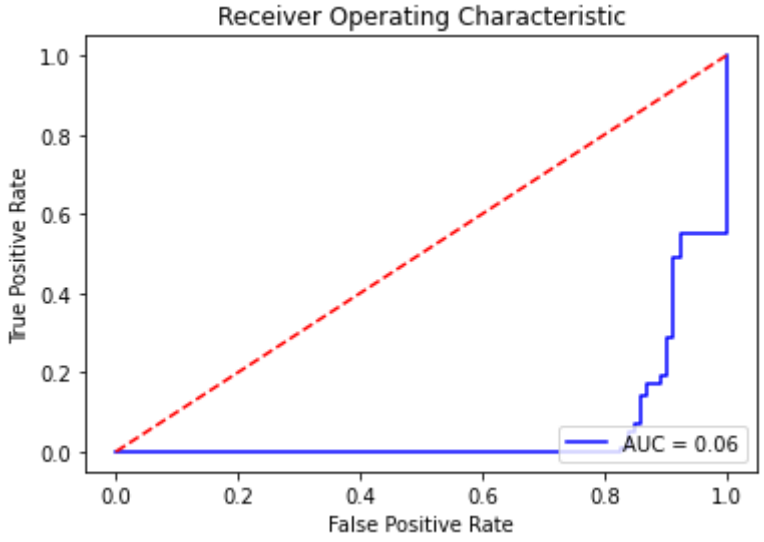
```
Epoch 42/50
22/22 [==============================] - 699s 32s/step - loss: 0.0388
Epoch 43/50
22/22 [==============================] - 716s 32s/step - loss: 0.0388
Epoch 44/50
22/22 [==============================] - 713s 32s/step - loss: 0.0388
Epoch 45/50
22/22 [==============================] - 703s 32s/step - loss: 0.0387
Epoch 46/50
22/22 [==============================] - 713s 32s/step - loss: 0.0387
Epoch 47/50
22/22 [==============================] - 704s 32s/step - loss: 0.0387
Epoch 48/50
22/22 [==============================] - 705s 32s/step - loss: 0.0387
Epoch 49/50
22/22 [==============================] - 708s 32s/step - loss: 0.0387
Epoch 50/50
22/22 [==============================] - 696s 32s/step - loss: 0.0387
got model
(200, 227, 227, 1)
got data
```

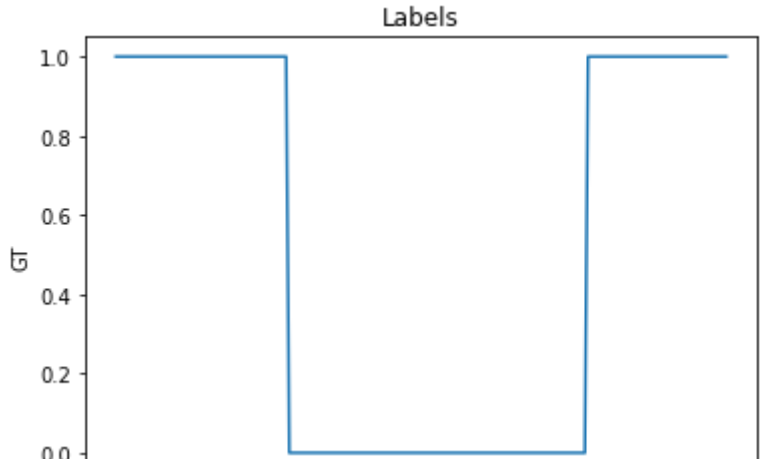


In [11]:

```
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:  0.05859117840684662
EER:  0.8602150537634409
EER THRESHOLD:  0.9806001651416959
Optimal threshold value is: 2.0
```

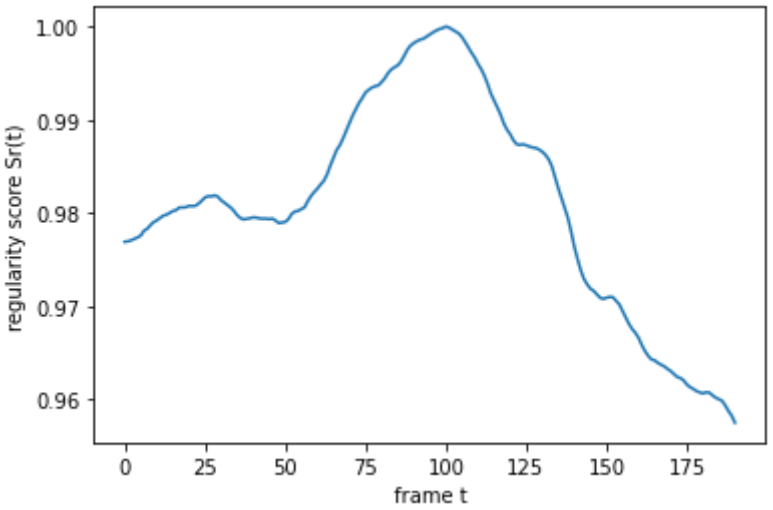

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
AUC = 0.06
0.0
0.2
0.4
0.6
0.8
1.0


Labels
GT
1.0
0.8
0.6
0.4
0.2
0.0

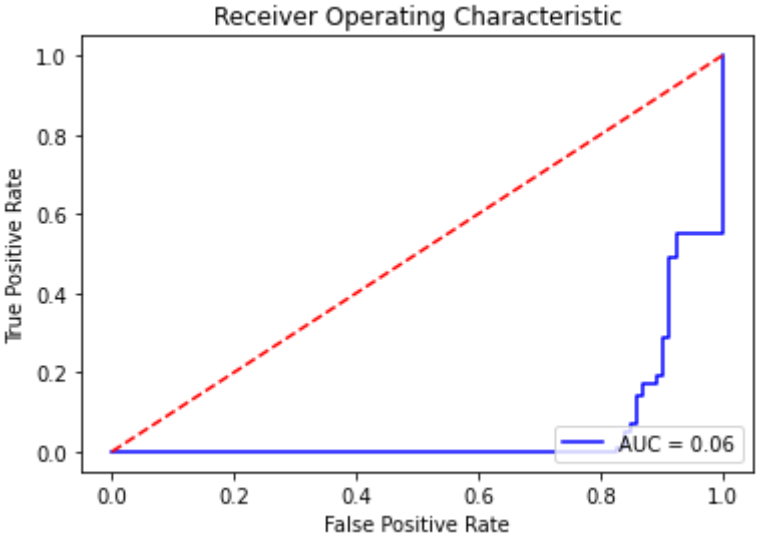
In [12]:

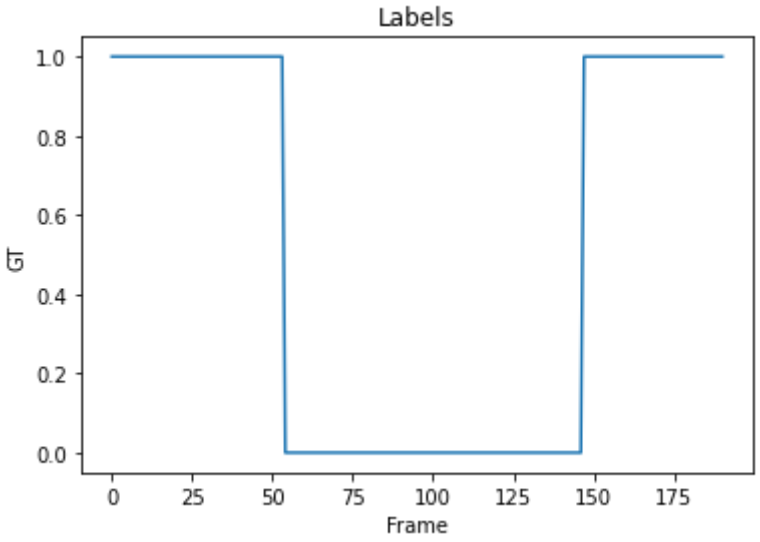
```
from os import listdir
from os.path import isfile, join, isdir


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
```

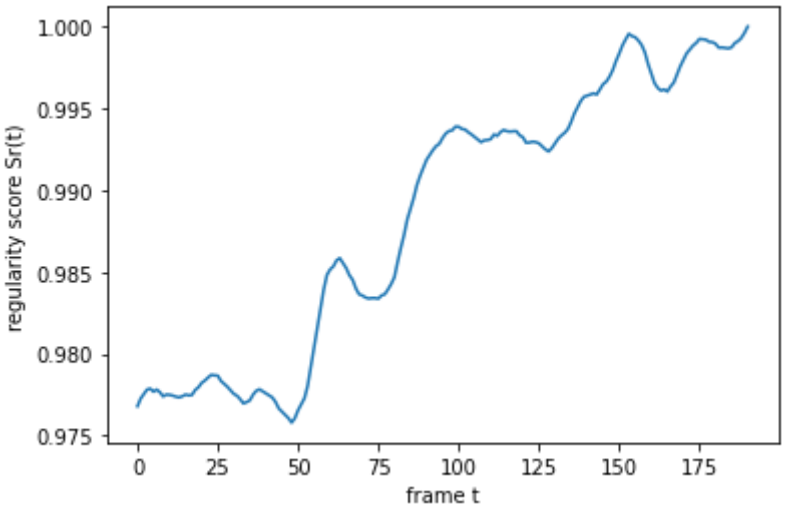



```
AUC:  0.05859117840684662
EER:  0.8602150537634409
EER THRESHOLD:  0.9806001651416959
Optimal threshold value is: 2.0
```

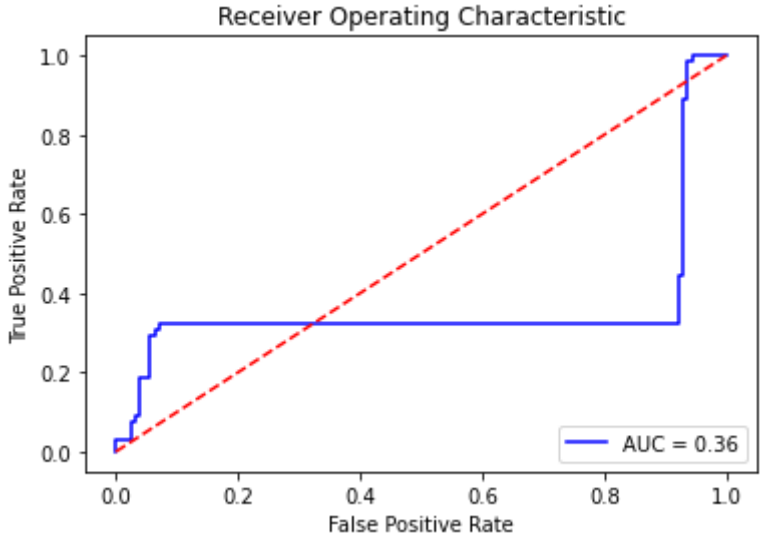


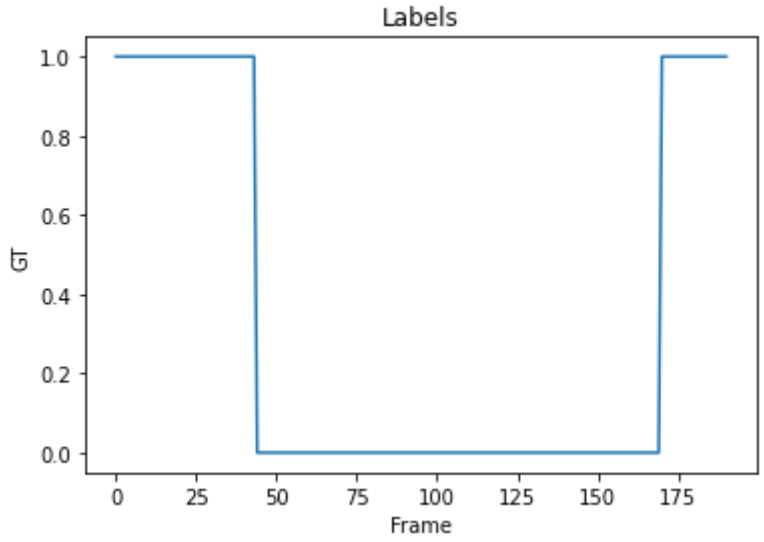



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test002
GT:  2
got model
(200, 227, 227, 1)
got data
```

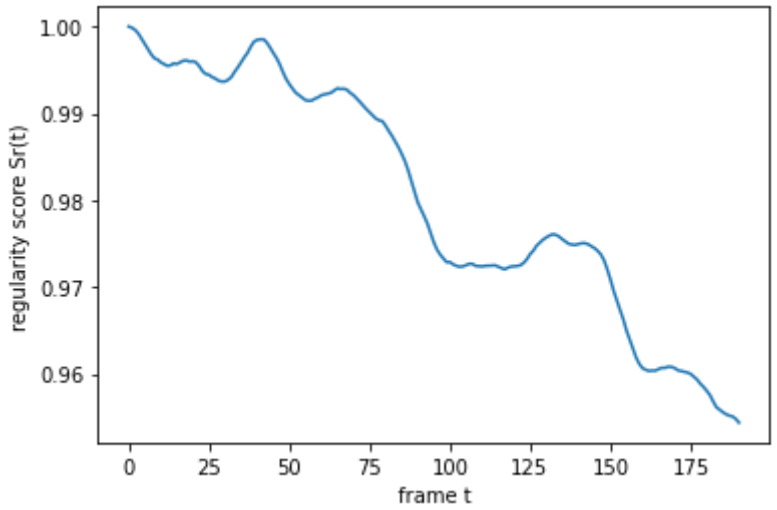



```
AUC:  0.3581196581196581
EER:  0.9206349206349206
EER THRESHOLD:  0.97921546911212
Optimal threshold value is: 0.9979748987510687
```

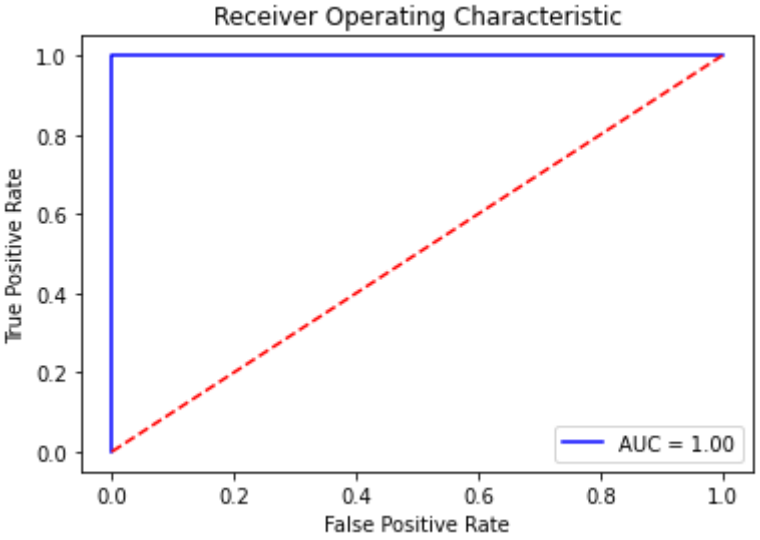


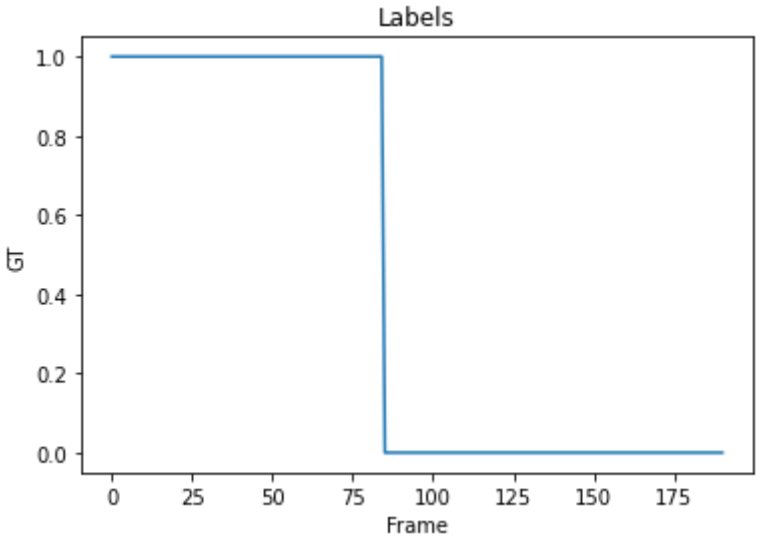



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test003
GT:  3
got model
(200, 227, 227, 1)
got data
```

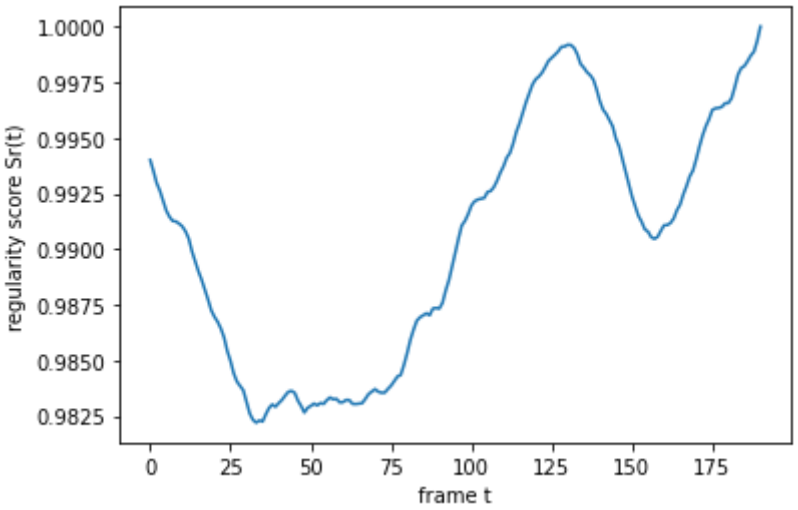



```
AUC:  1.0
EER:  0.0
EER THRESHOLD:  0.9861049503932645
Optimal threshold value is: 0.9861049503932645
```

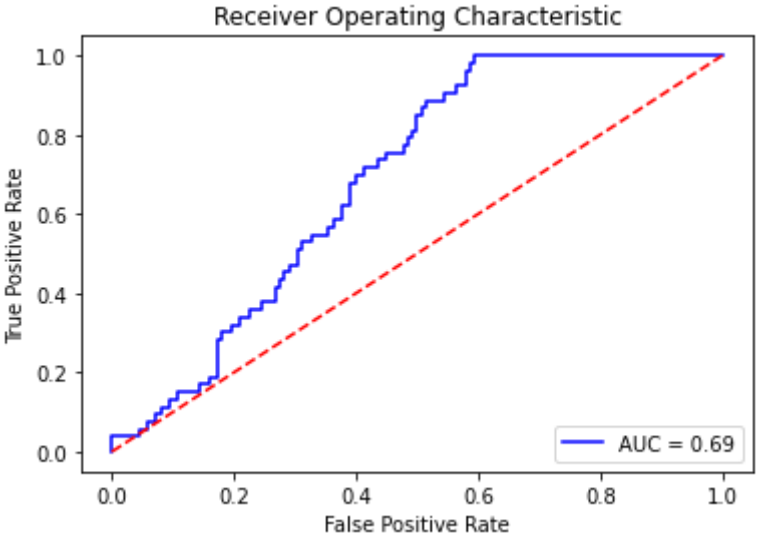


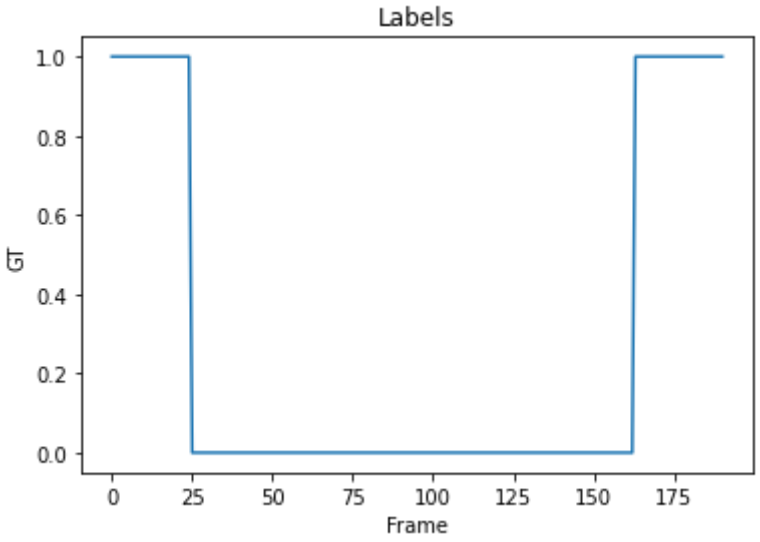



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test004
GT:  4
got model
(200, 227, 227, 1)
got data
```

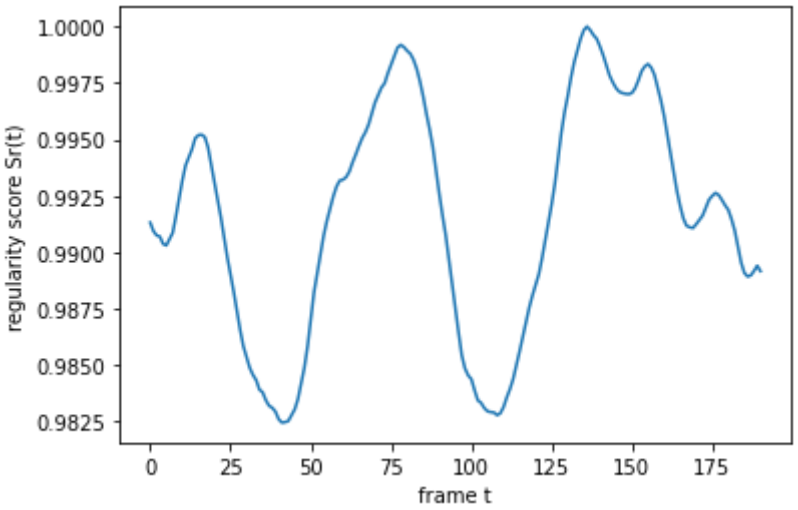



```
AUC:  0.6870385561936012
EER:  0.37681159420289856
EER THRESHOLD:  0.991729399686894
Optimal threshold value is: 0.9854214559170957
```

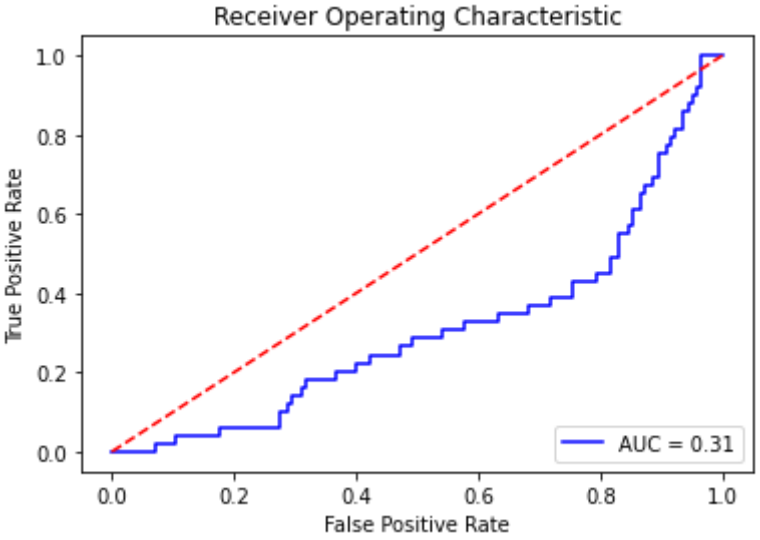


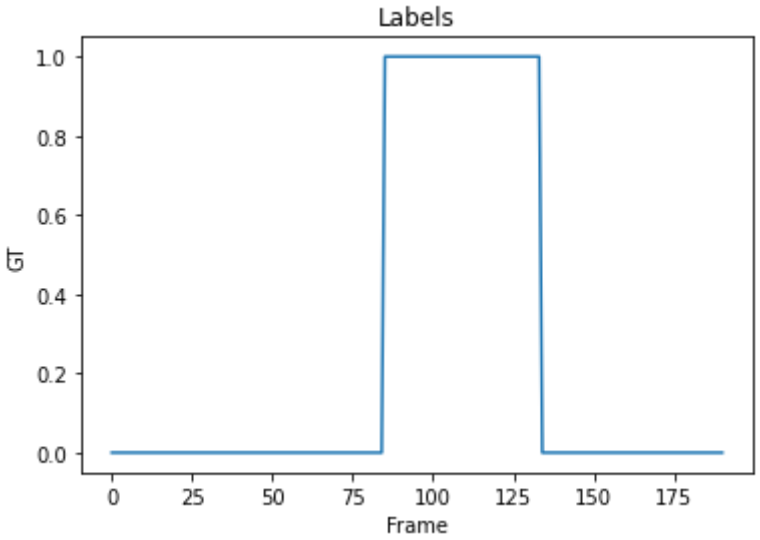



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test005
GT:  5
got model
(200, 227, 227, 1)
got data
```

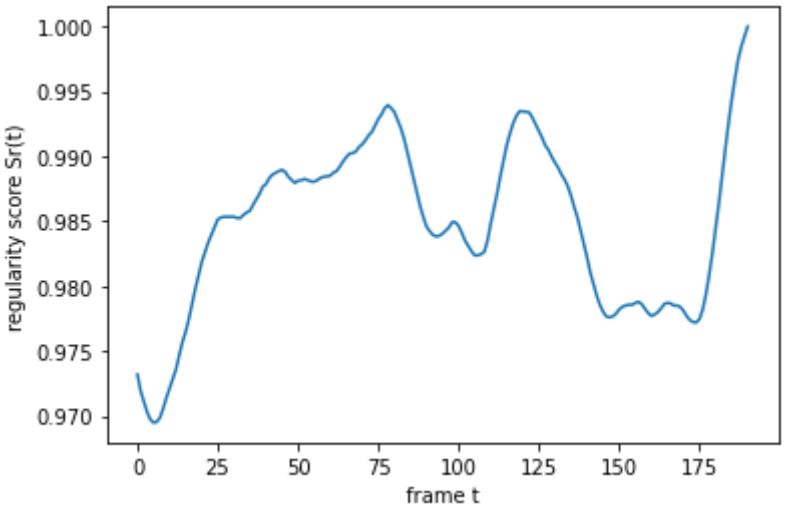



```
AUC:  0.3079908019545846
EER:  0.6338028169014085
EER THRESHOLD:  0.9912778917704352
Optimal threshold value is: 0.9827871760603824
```

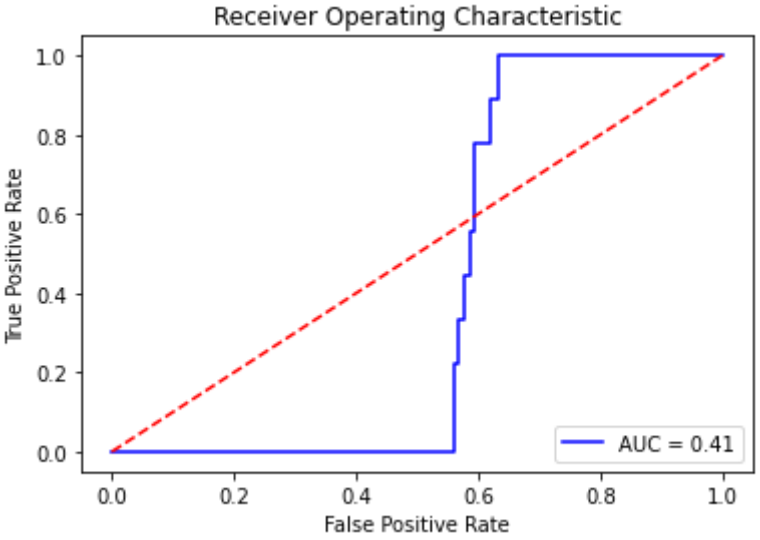


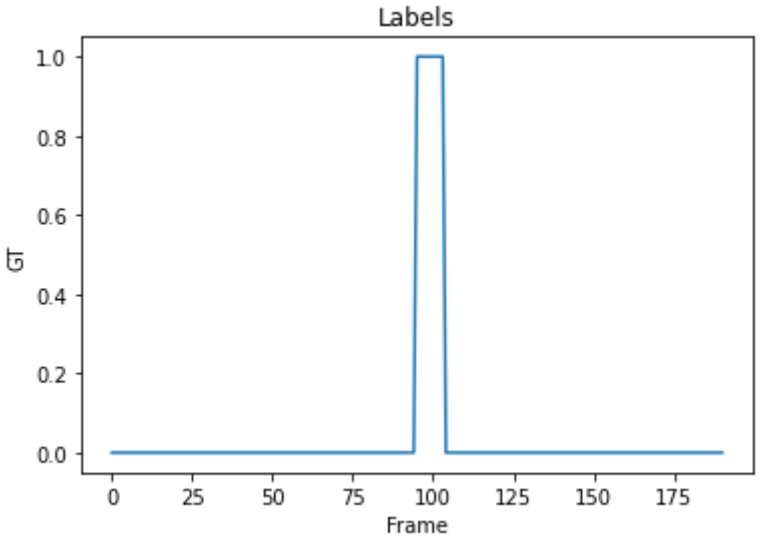



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test006
GT:  6
got model
(200, 227, 227, 1)
got data
```

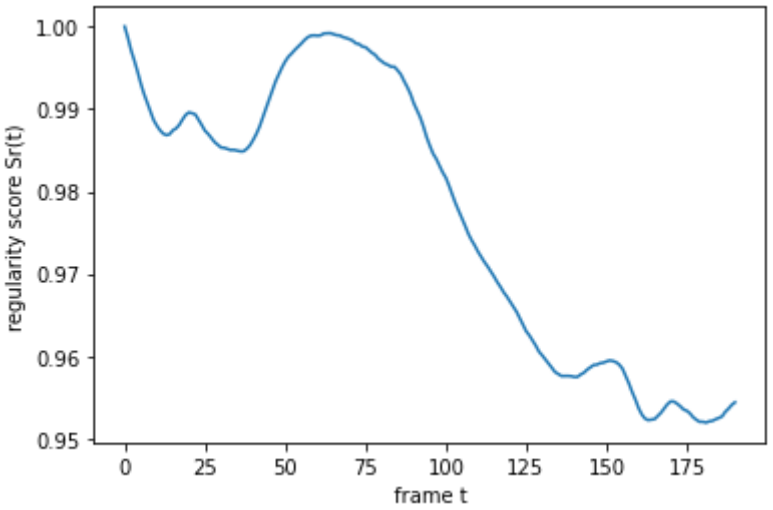



```
AUC:  0.4120879120879121
EER:  0.5769230769230769
EER THRESHOLD:  0.9845281063519282
Optimal threshold value is: 0.9831158886436443
```

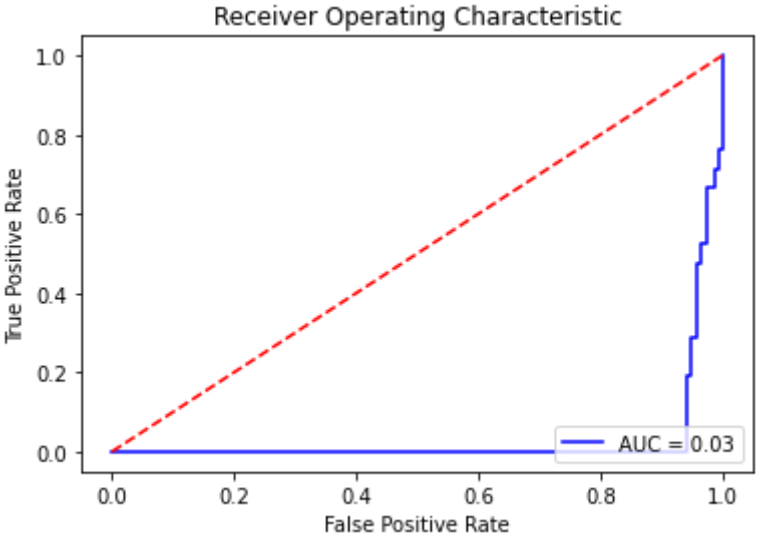


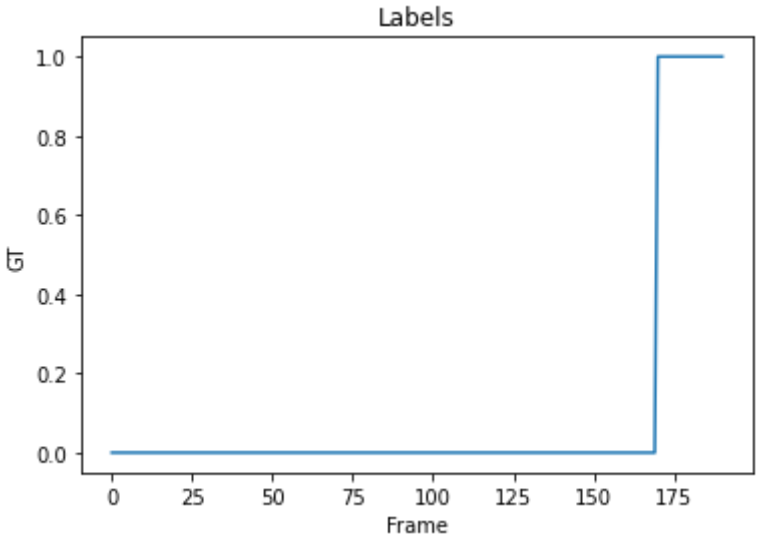



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test007
GT:  7
got model
(200, 227, 227, 1)
got data
```

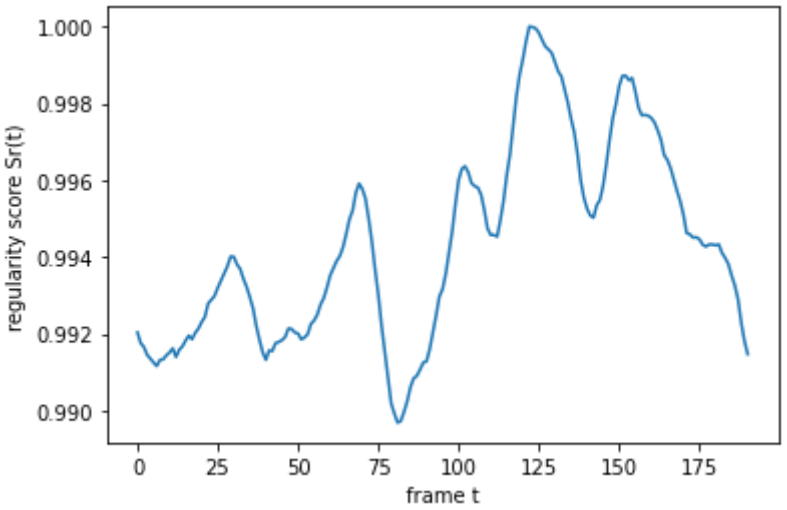



```
AUC:  0.029971988795518205
EER:  0.9411764705882353
EER THRESHOLD:  0.9547519334549605
Optimal threshold value is: 2.0
```

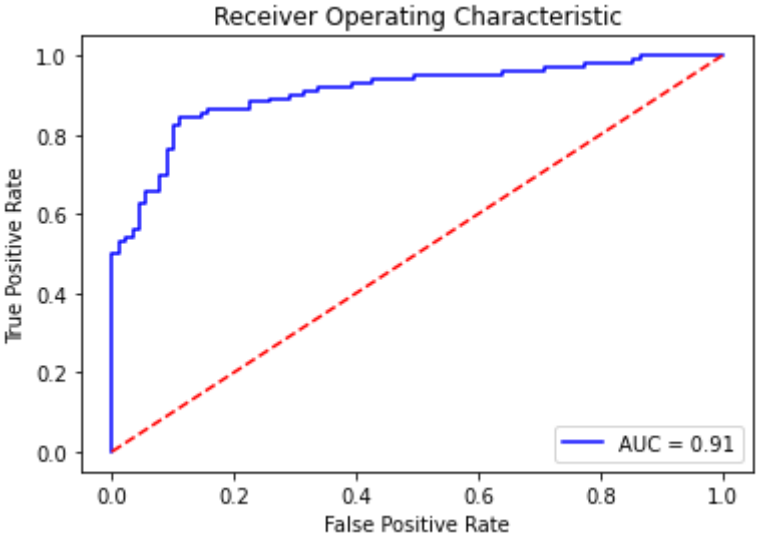


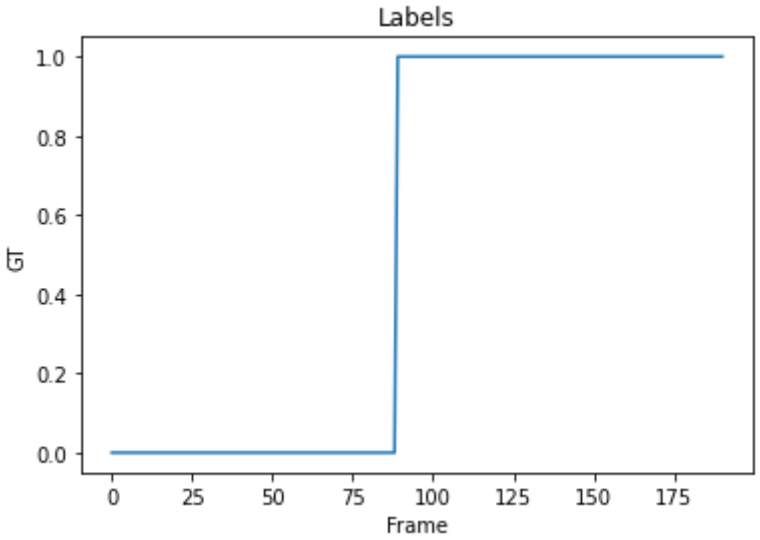



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test008
GT:  8
got model
(200, 227, 227, 1)
got data
```

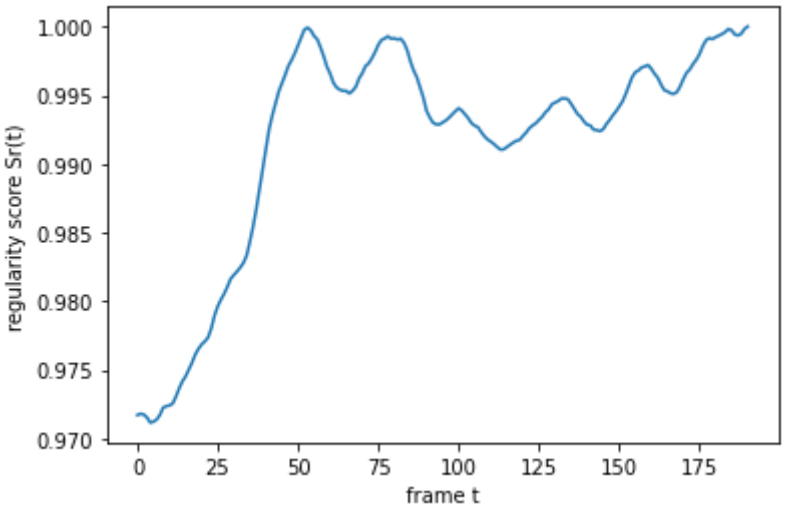



```
AUC:  0.9069178233090989
EER:  0.14606741573033707
EER THRESHOLD:  0.9939777905870432
Optimal threshold value is: 0.9941092076846275
```

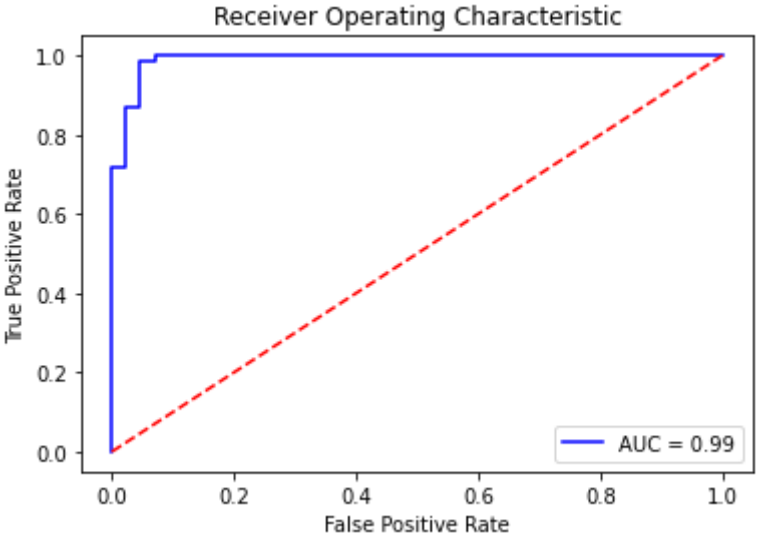


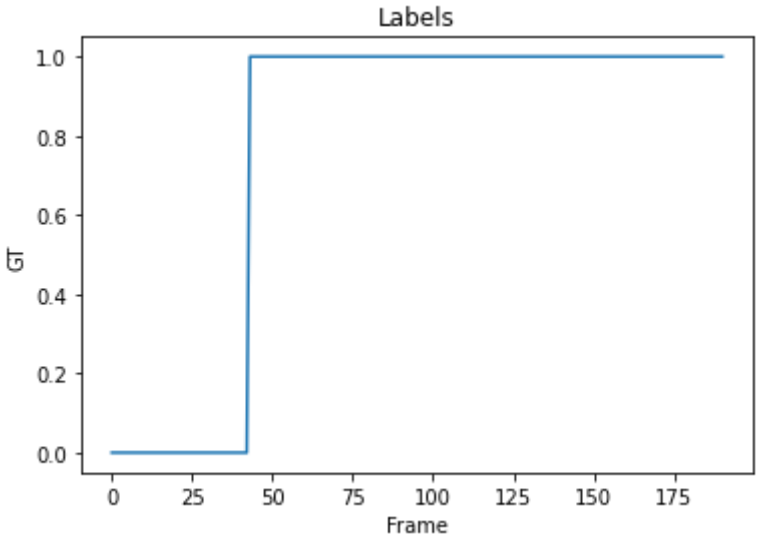



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test009
GT:  9
got model
(200, 227, 227, 1)
got data
```

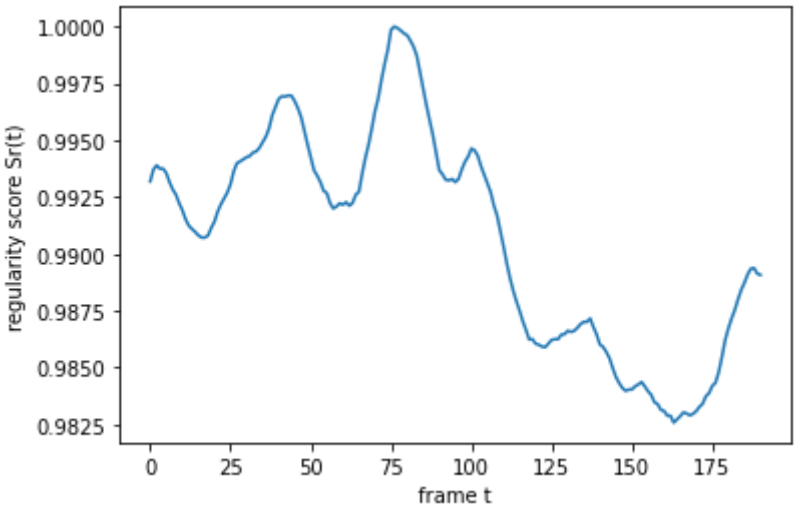



```
AUC:  0.990100565681961
EER:  0.046511627906976744
EER THRESHOLD:  0.9912249715109174
Optimal threshold value is: 0.9912249715109174
```

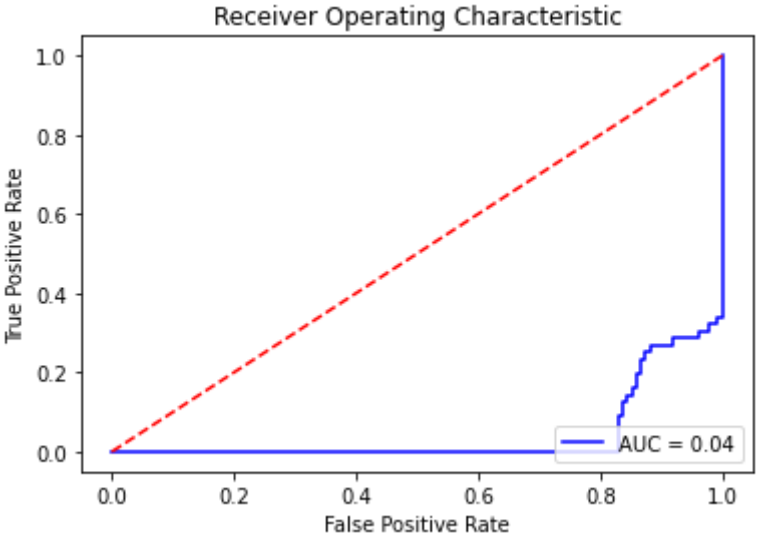


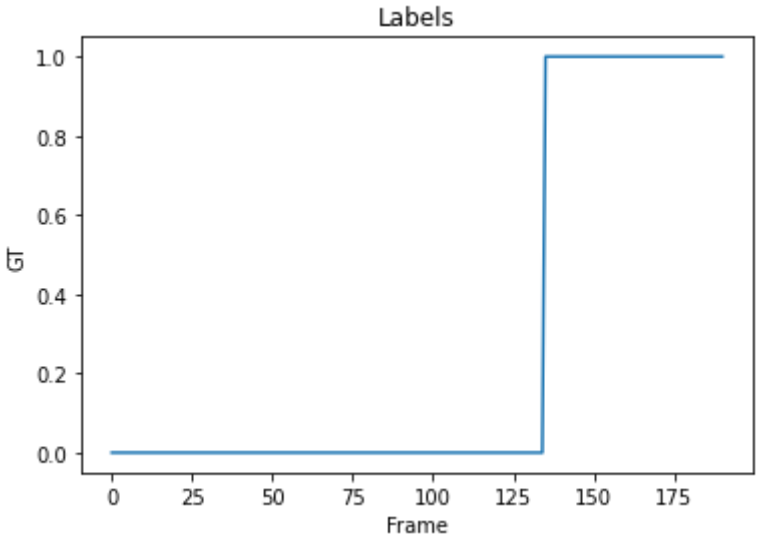



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test010
GT:  10
got model
(200, 227, 227, 1)
got data
```

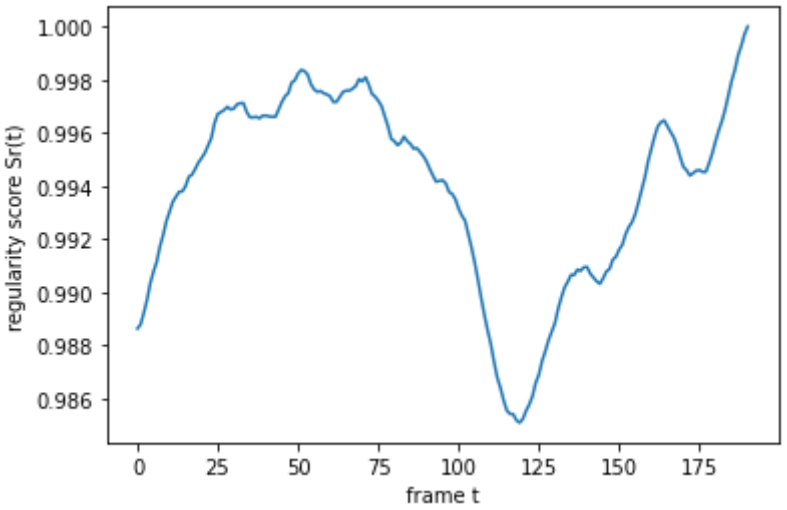



```
AUC:  0.04325396825396825
EER:  0.8518518518518519
EER THRESHOLD:  0.9879202084898209
Optimal threshold value is: 2.0
```

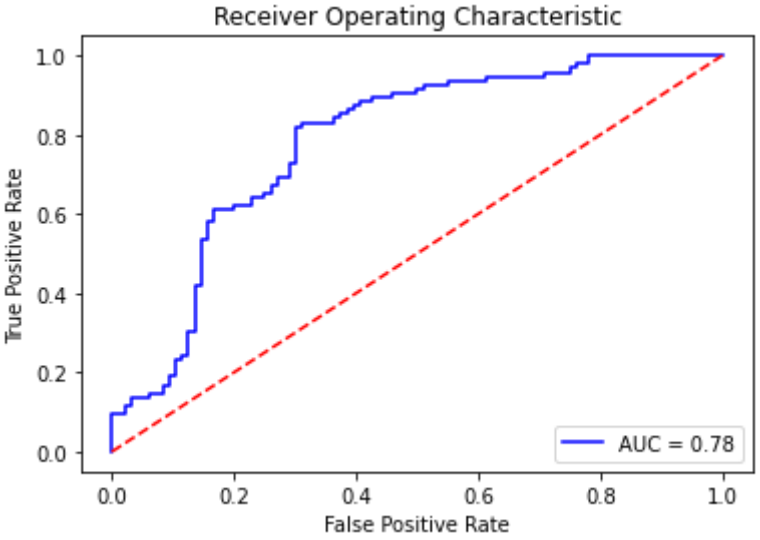


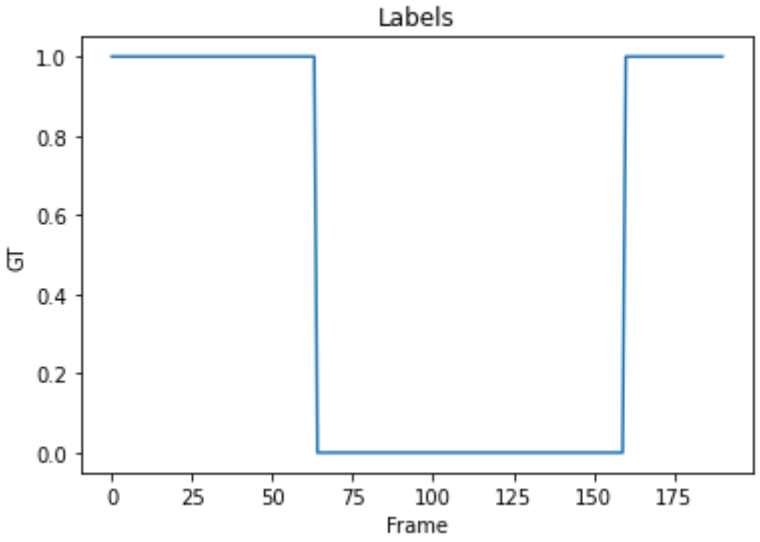



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test011
GT:  11
got model
(200, 227, 227, 1)
got data
```

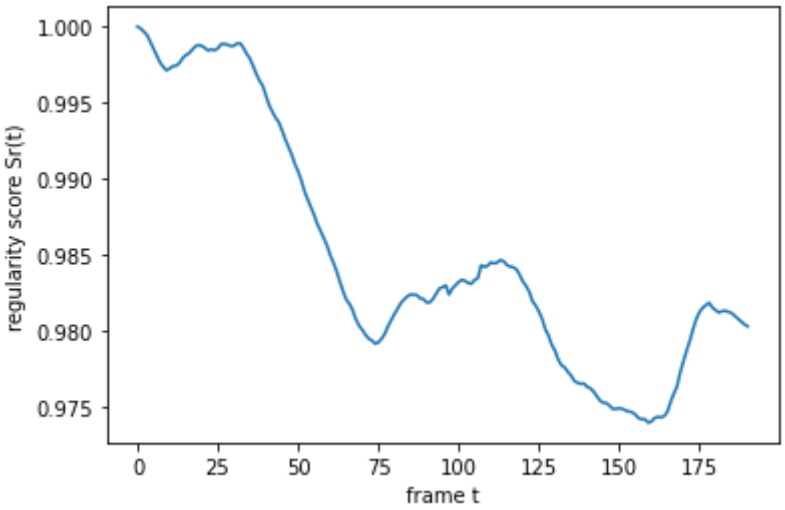



```
AUC:  0.7824561403508772
EER:  0.2916666666666667
EER THRESHOLD:  0.9949236380316783
Optimal threshold value is: 0.994345595379242
```

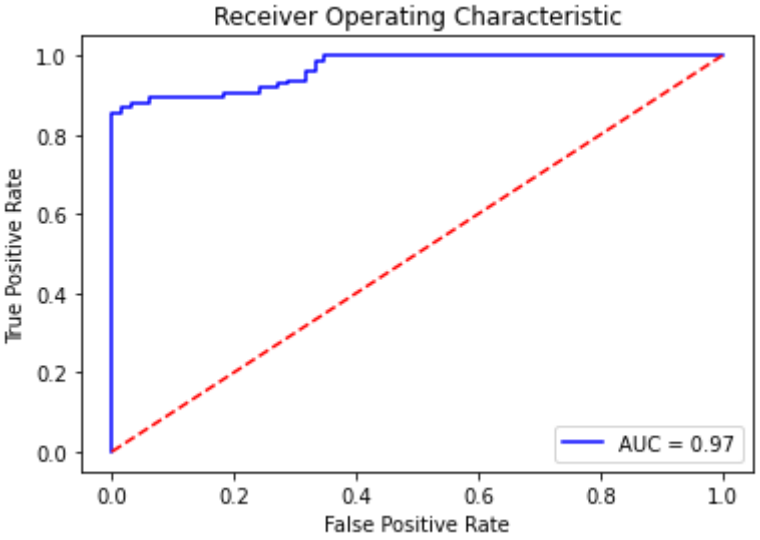


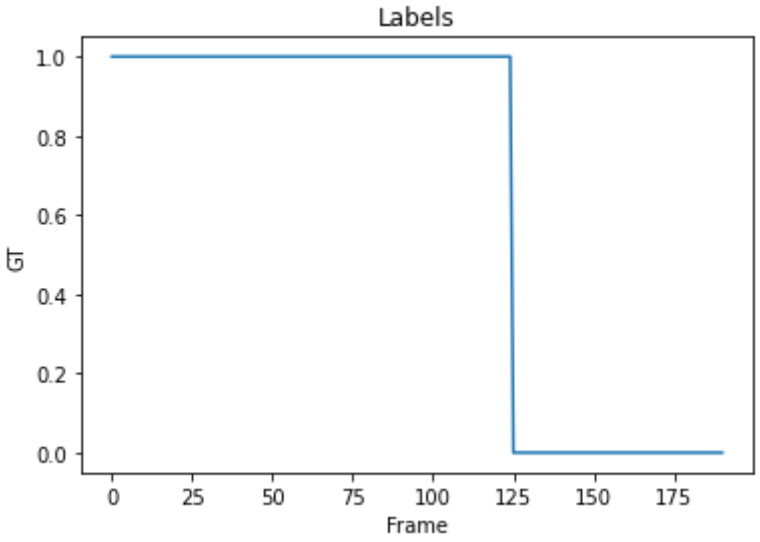



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test012
GT:  12
got model
(200, 227, 227, 1)
got data
```

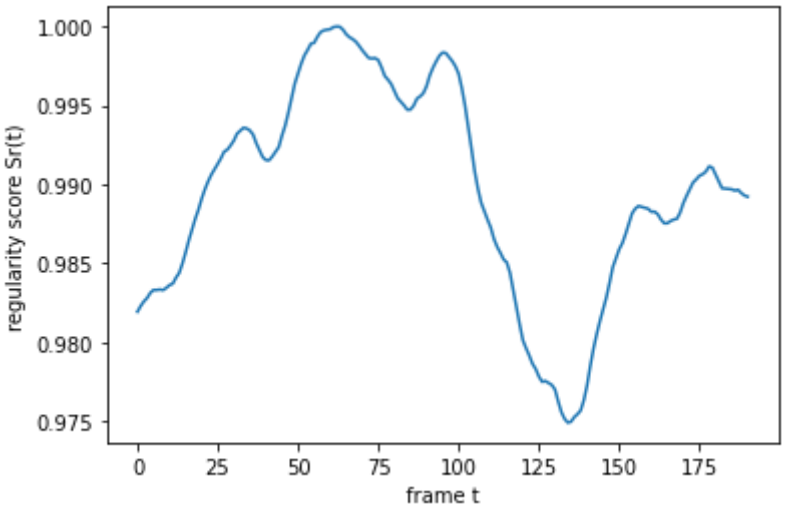



```
AUC:  0.9675151515151514
EER:  0.06060606060606061
EER THRESHOLD:  0.9813644807277019
Optimal threshold value is: 0.9817748049852832
```

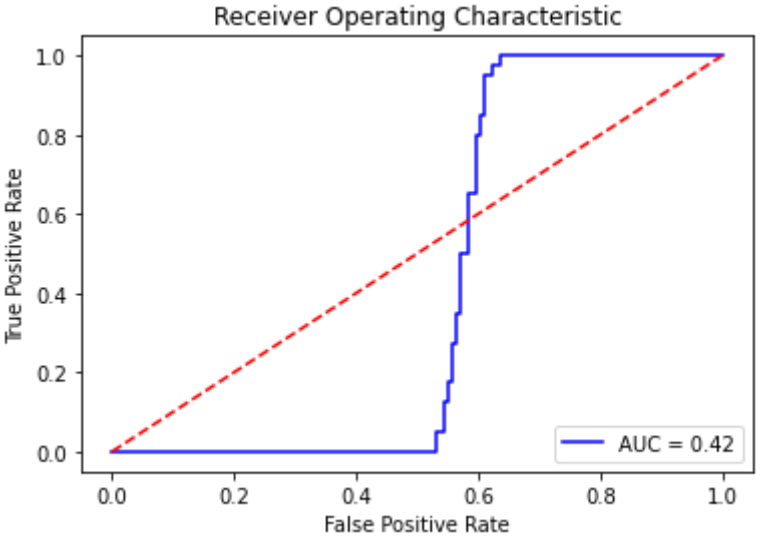


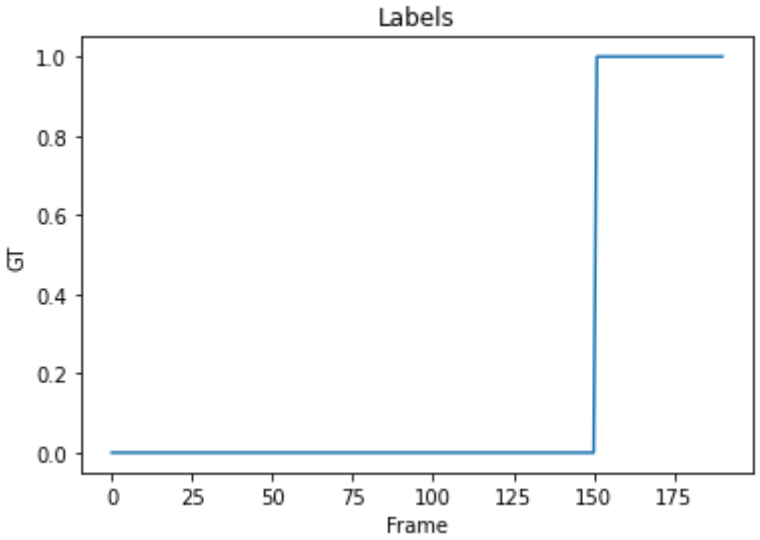



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test013
GT:  13
got model
(200, 227, 227, 1)
got data
```

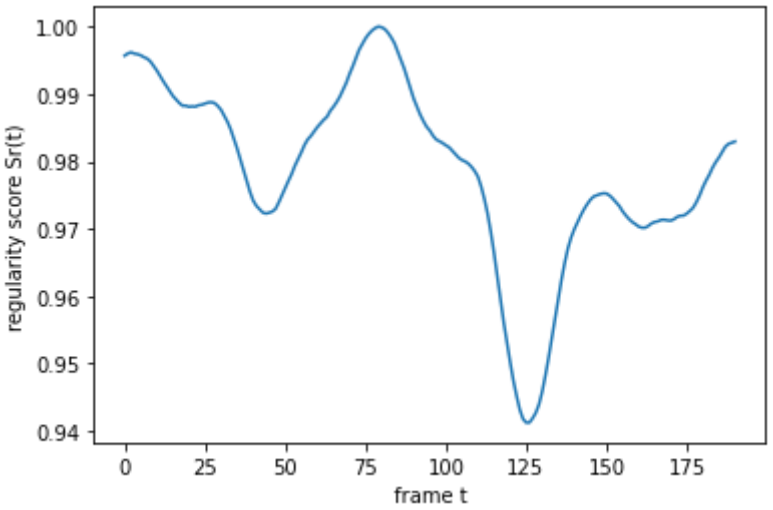



```
AUC:  0.422682119205298
EER:  0.5695364238410596
EER THRESHOLD:  0.9892185542094898
Optimal threshold value is: 0.9862840362760616
```

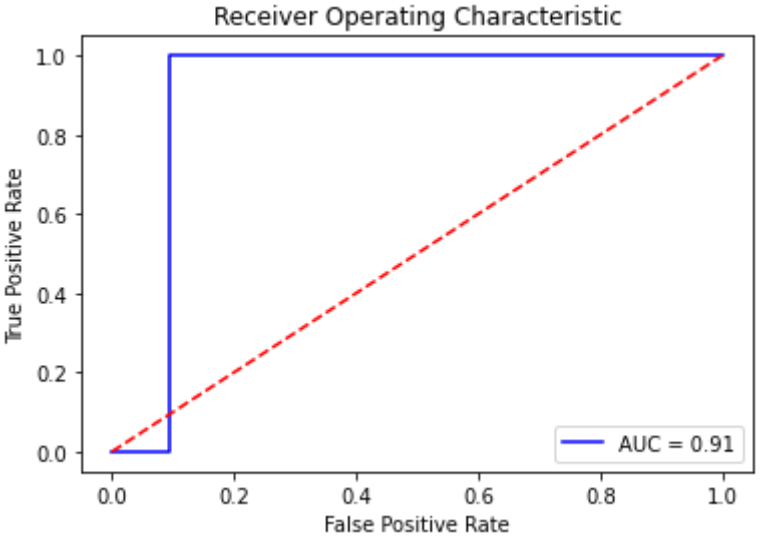


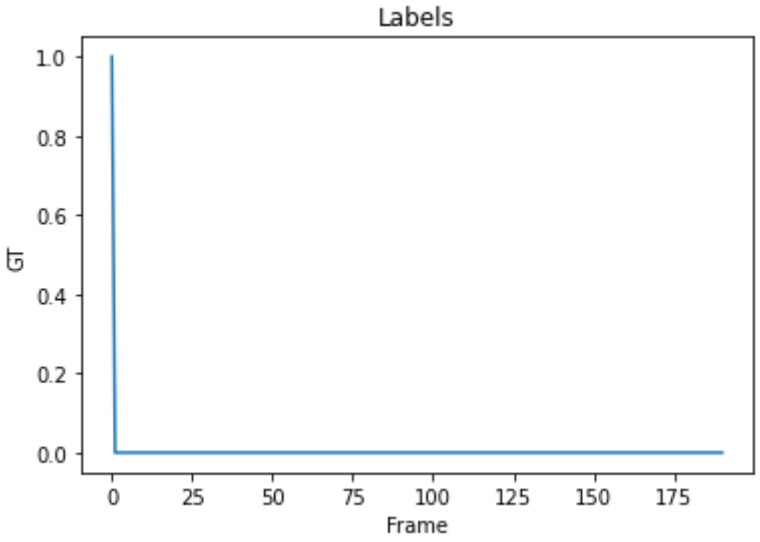



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test014
GT:  14
got model
(200, 227, 227, 1)
got data
```

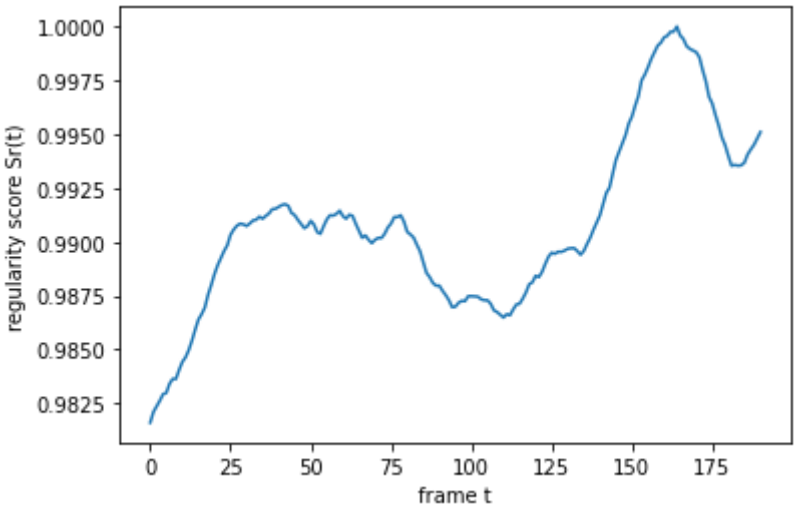



```
AUC:  0.9052631578947369
EER:  0.09473684210526316
EER THRESHOLD:  0.995684673118976
Optimal threshold value is: 0.995684673118976
```

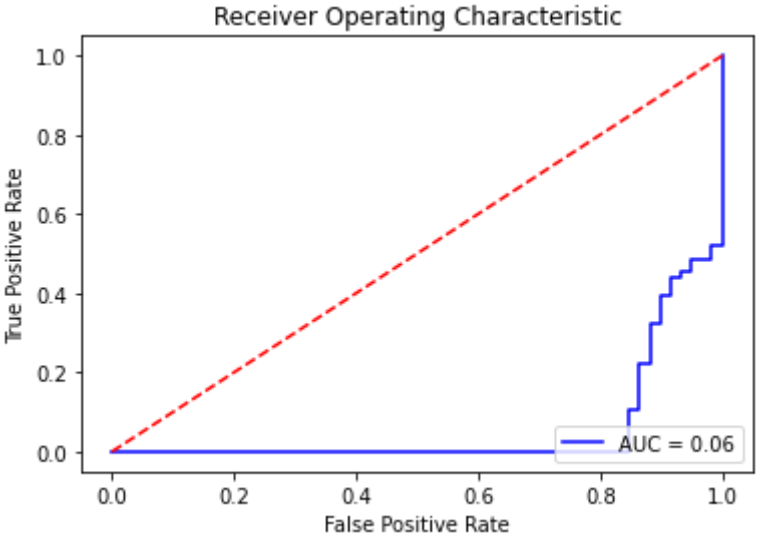


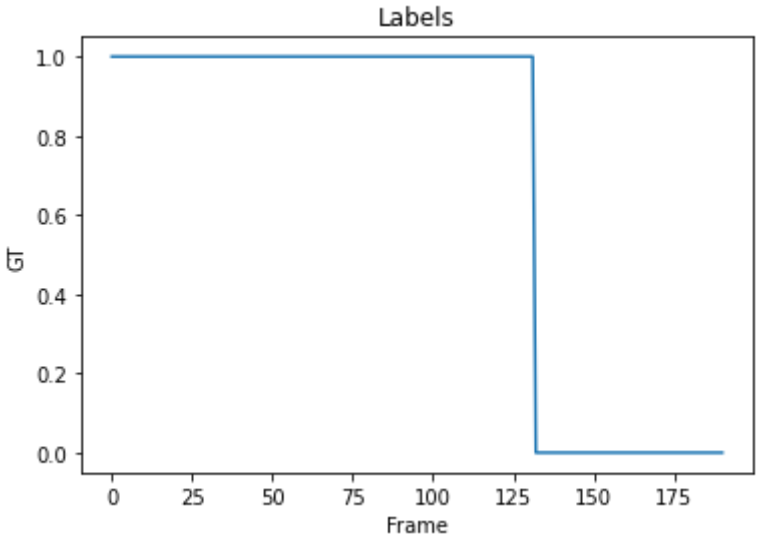



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test015
GT:  15
got model
(200, 227, 227, 1)
got data
```

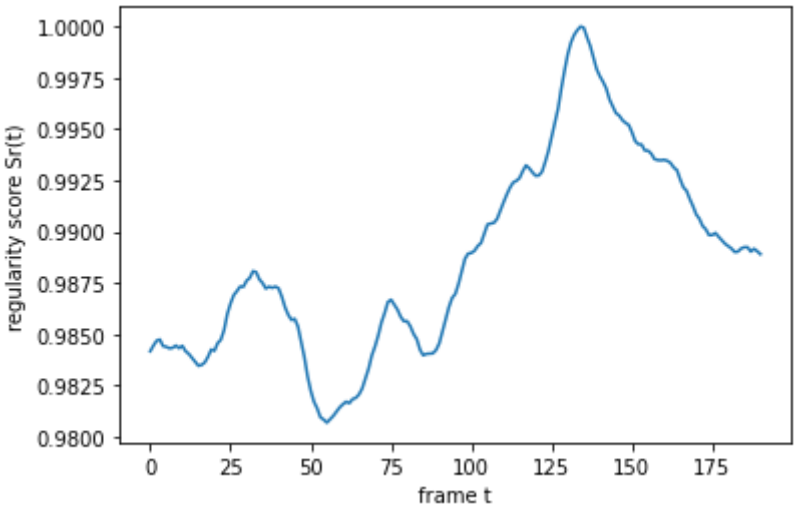



```
AUC:  0.05816640986132511
EER:  0.864406779661017
EER THRESHOLD:  0.991228686457753
Optimal threshold value is: 2.0
```

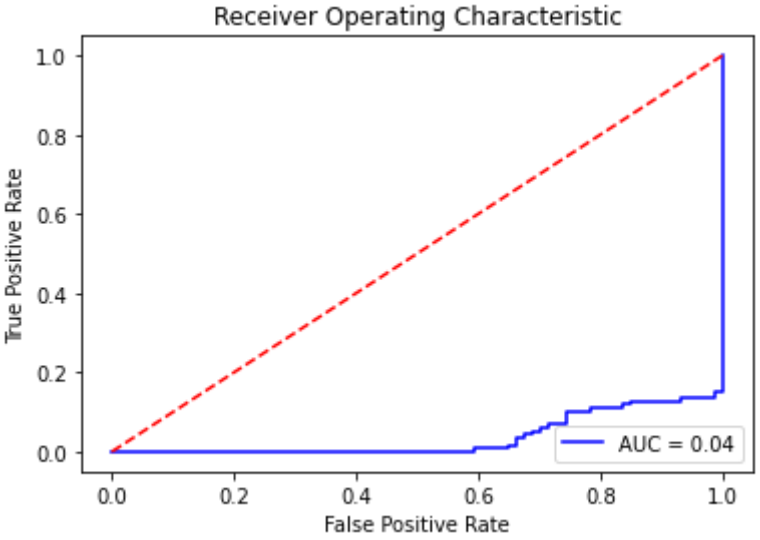


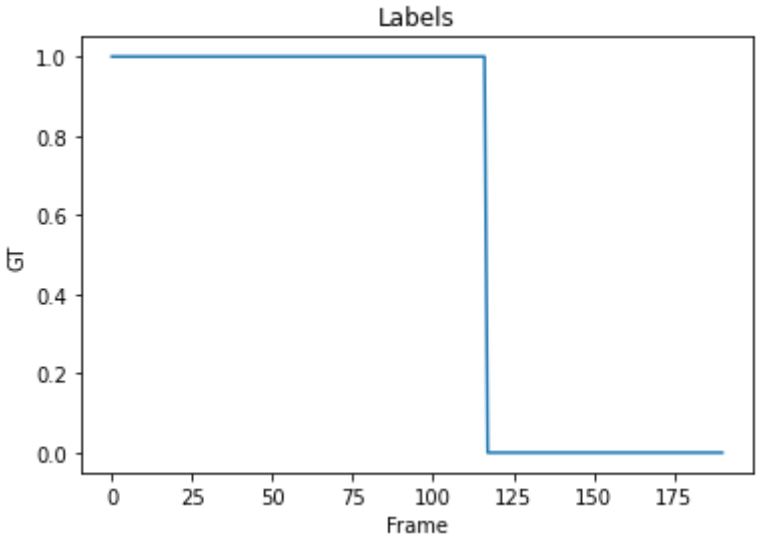



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test016
GT:  16
got model
(200, 227, 227, 1)
got data
```

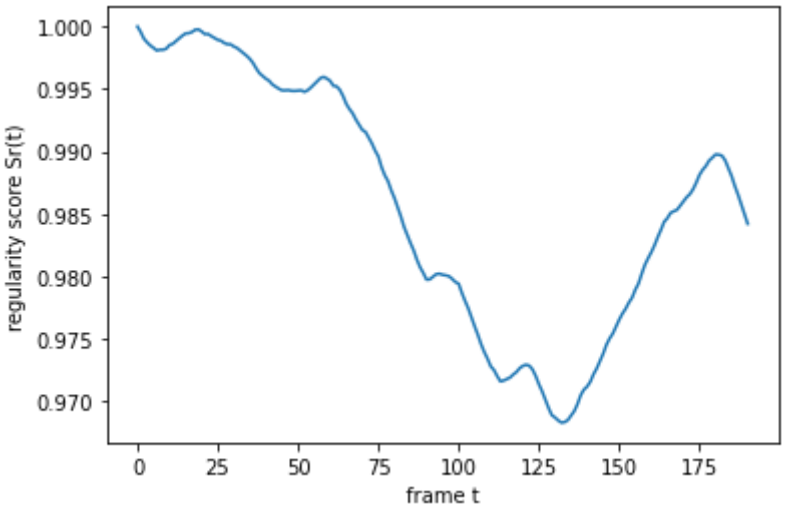



```
AUC:  0.036729036729036726
EER:  0.8513513513513513
EER THRESHOLD:  0.9893011205611911
Optimal threshold value is: 2.0
```

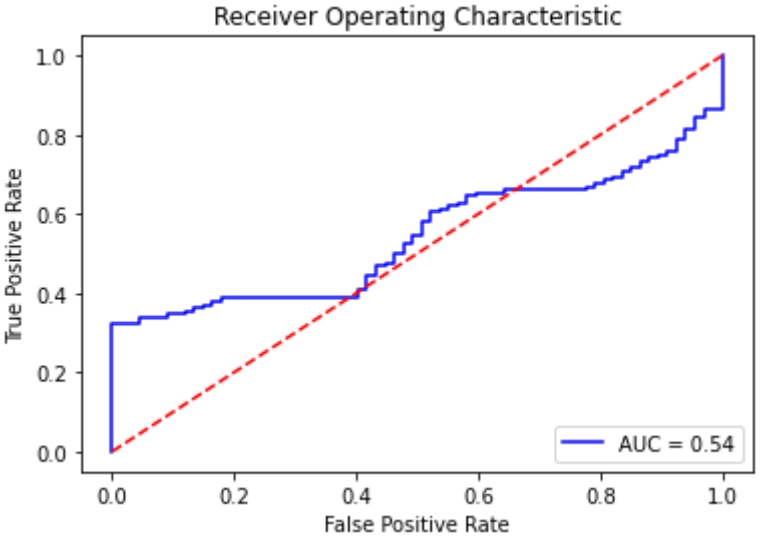


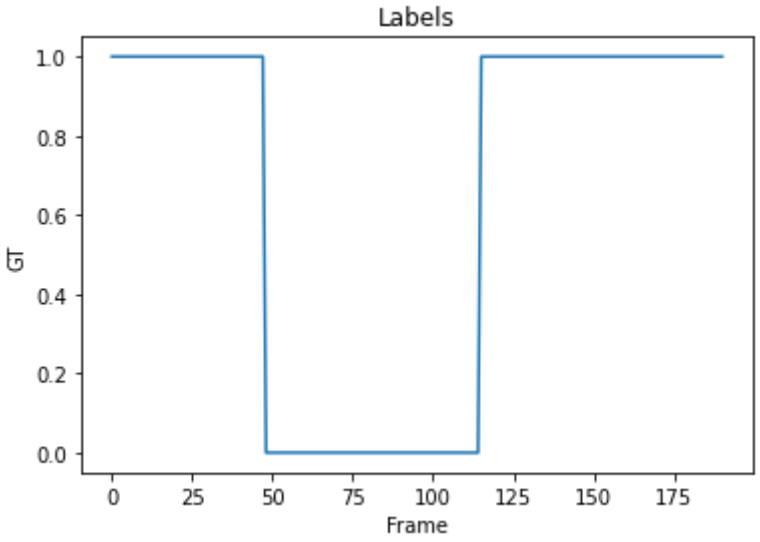



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test018
GT:  18
got model
(200, 227, 227, 1)
got data
```

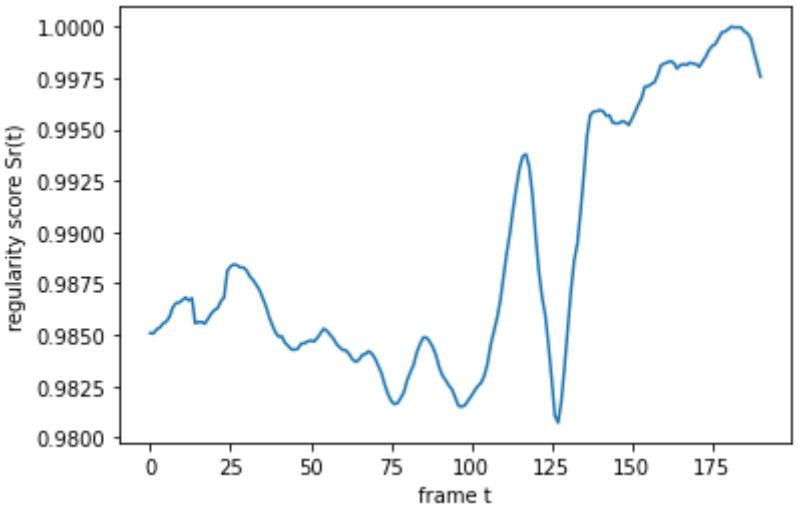



```
AUC:  0.5409244102070294
EER:  0.47761194029850745
EER THRESHOLD:  0.986334996858005
Optimal threshold value is: 0.9960344729298398
```

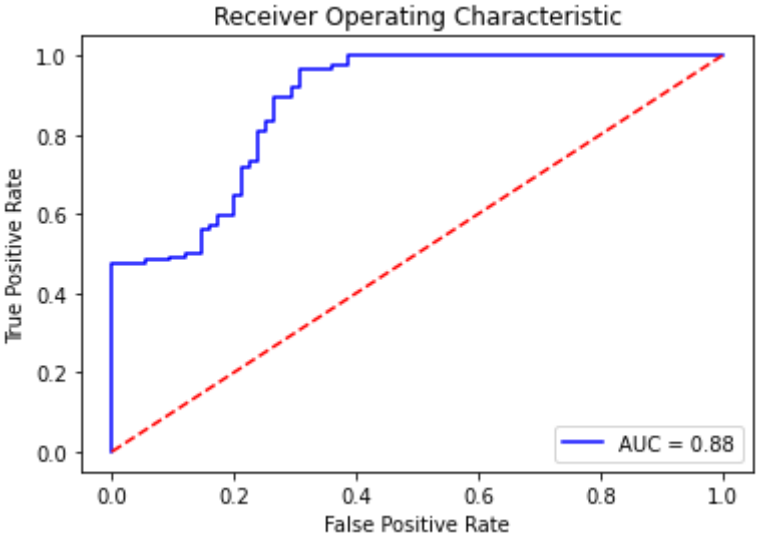


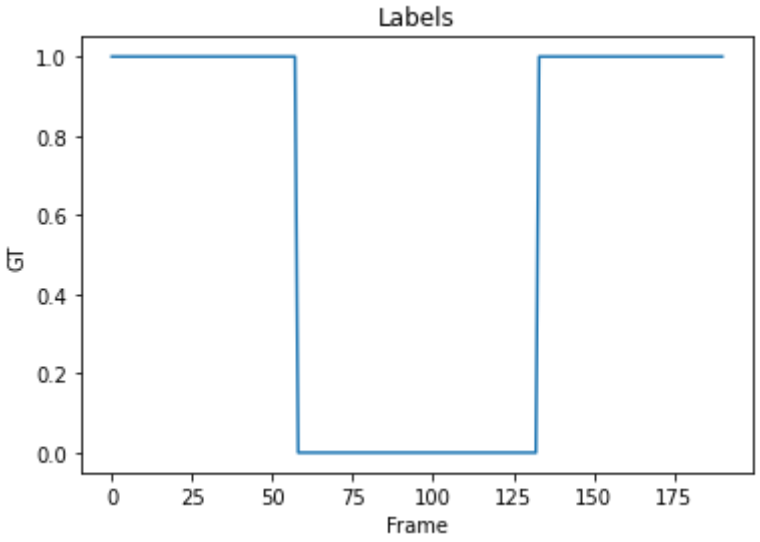



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test019
GT:  19
got model
(200, 227, 227, 1)
got data
```

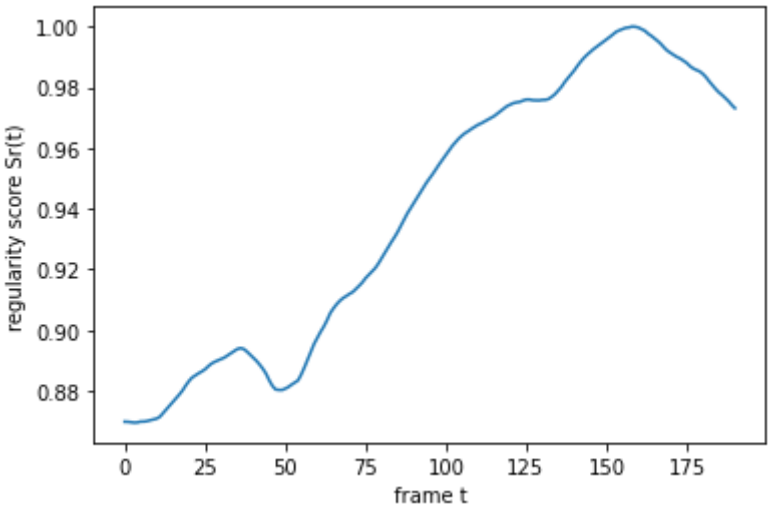



```
AUC:  0.8788505747126436
EER:  0.24
EER THRESHOLD:  0.9858158624717385
Optimal threshold value is: 0.9845698150775611
```

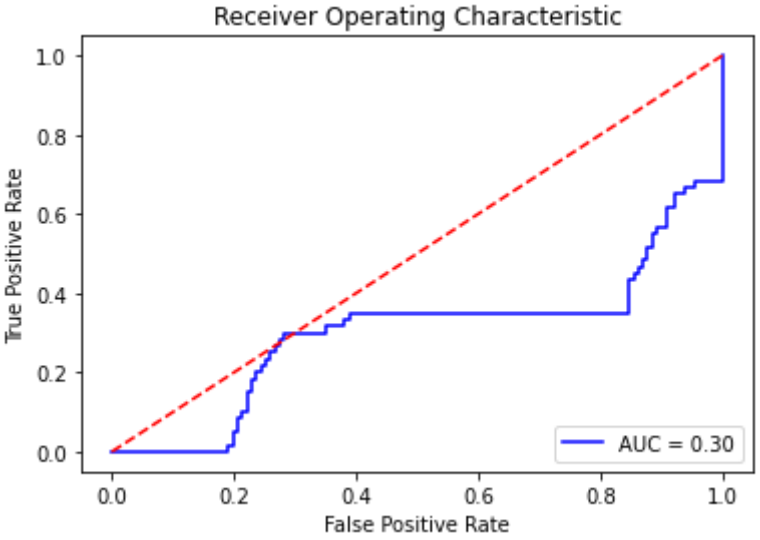


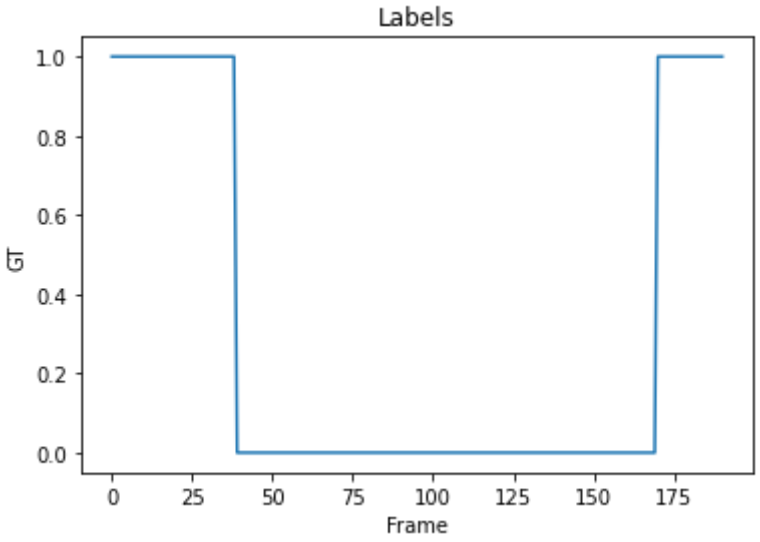



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test020
GT:  20
got model
(200, 227, 227, 1)
got data
```

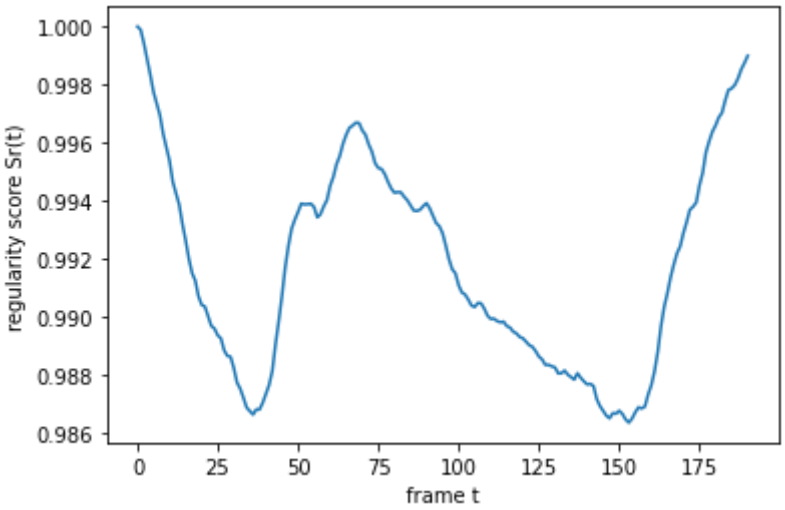



```
AUC:  0.30025445292620867
EER:  0.8473282442748091
EER THRESHOLD:  0.895541687461719
Optimal threshold value is: 0.9764827082817749
```

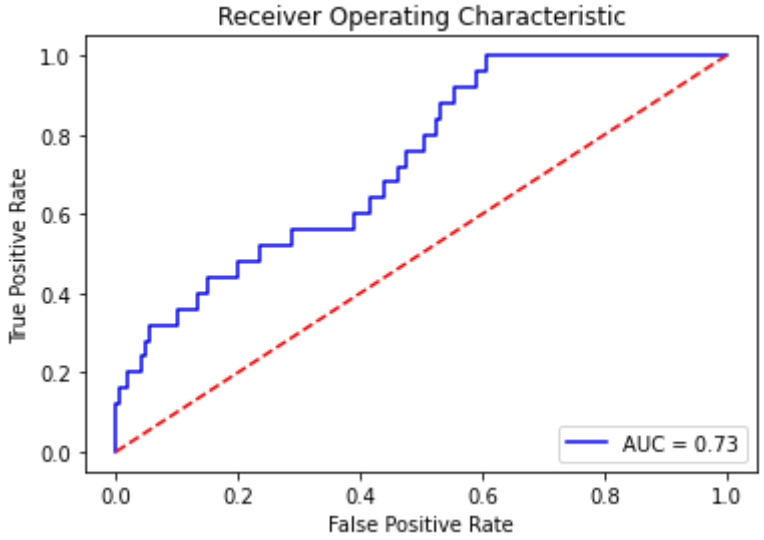


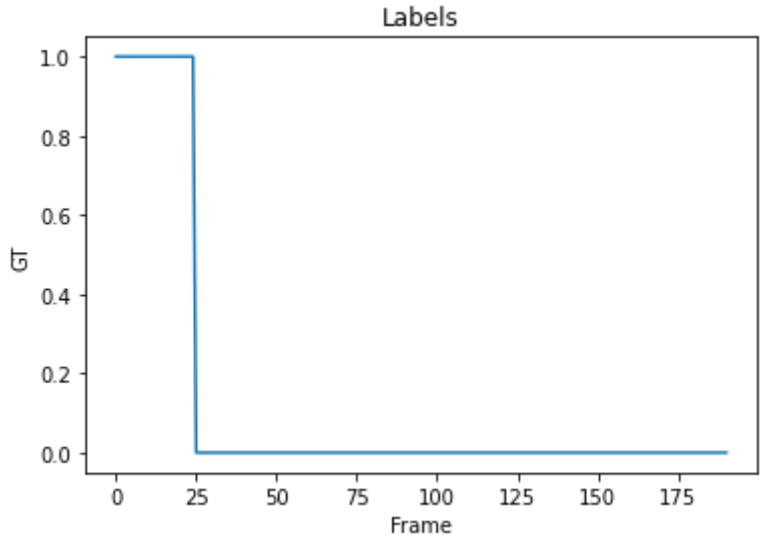



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test021
GT:  21
got model
(200, 227, 227, 1)
got data
```

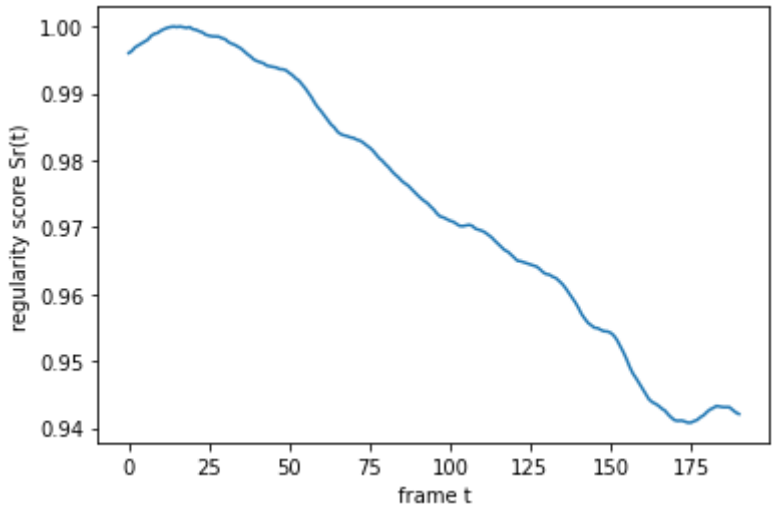



```
AUC:  0.7289156626506024
EER:  0.39156626506024095
EER THRESHOLD:  0.9931913710743429
Optimal threshold value is: 0.989574259853641
```

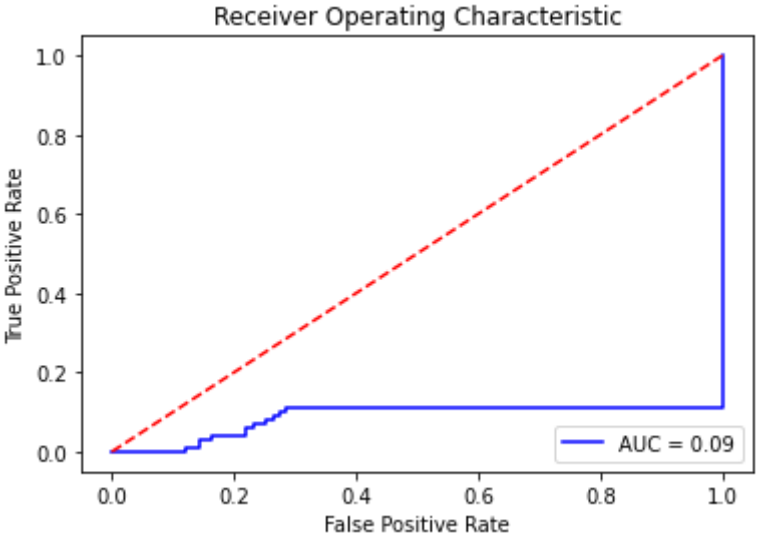


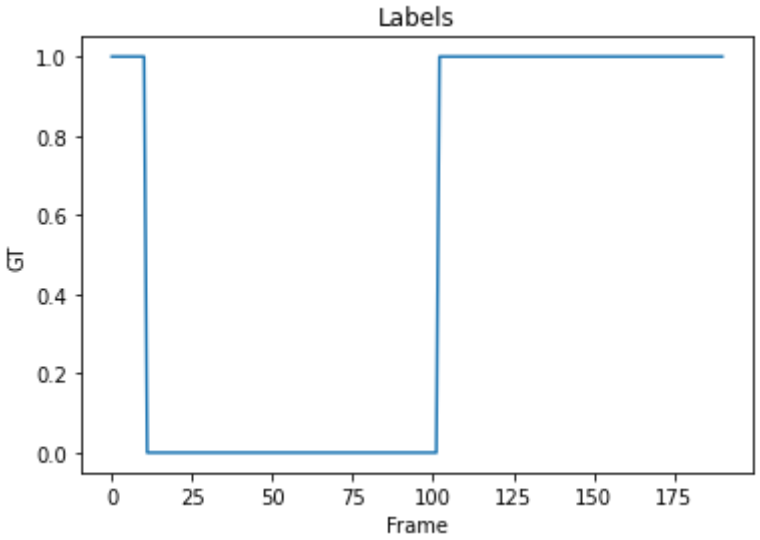



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test022
GT:  22
got model
(200, 227, 227, 1)
got data
```

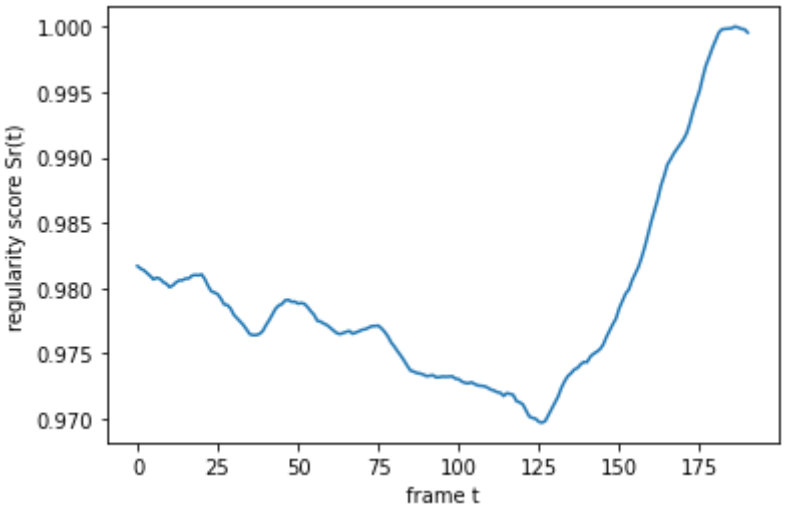



```
AUC:  0.08681318681318681
EER:  1.0
EER THRESHOLD:  0.970835745765403
Optimal threshold value is: 2.0
```

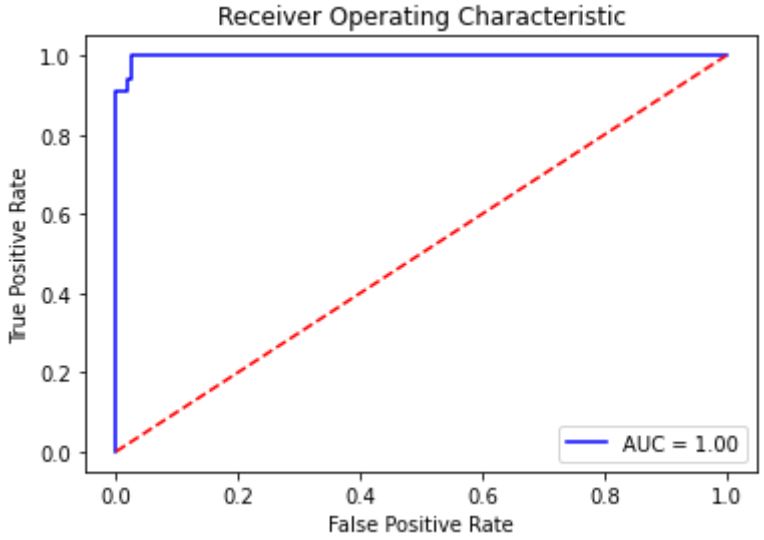


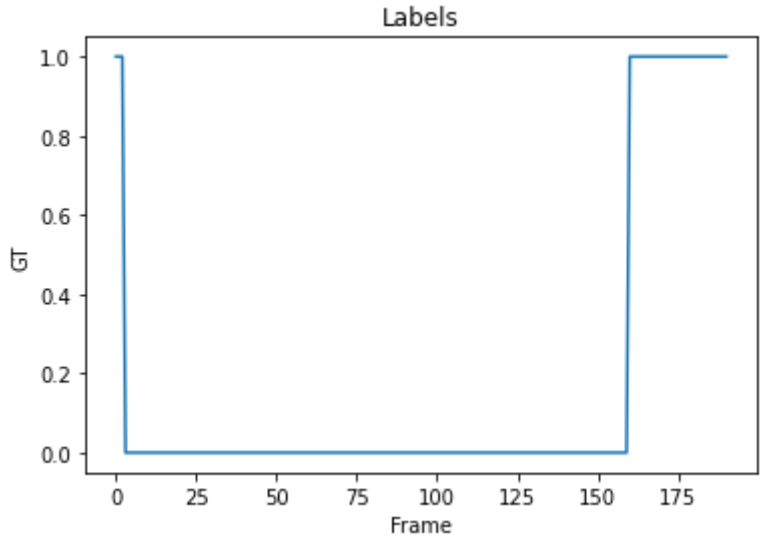



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test023
GT:  23
got model
(200, 227, 227, 1)
got data
```

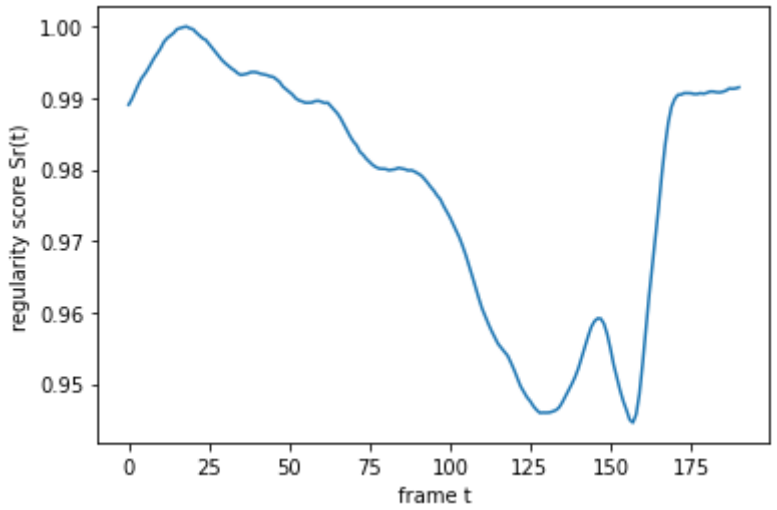



```
AUC:  0.997939303109779
EER:  0.025477707006369428
EER THRESHOLD:  0.9813667631665636
Optimal threshold value is: 0.9813667631665636
```

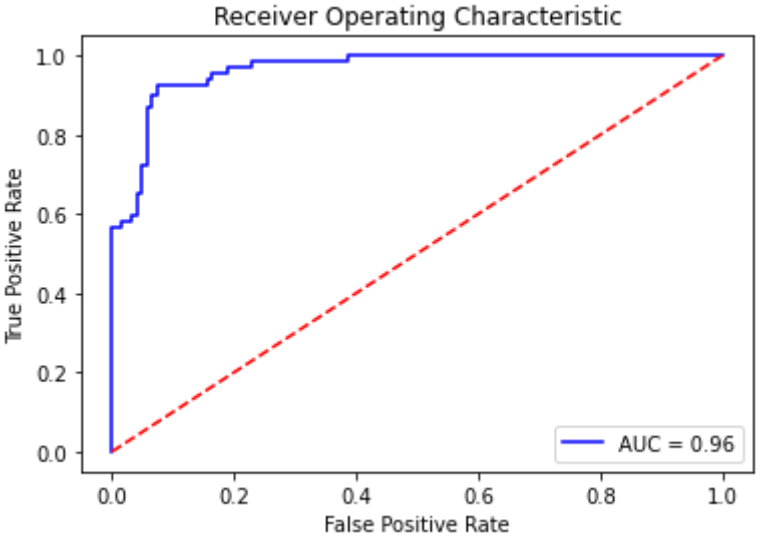


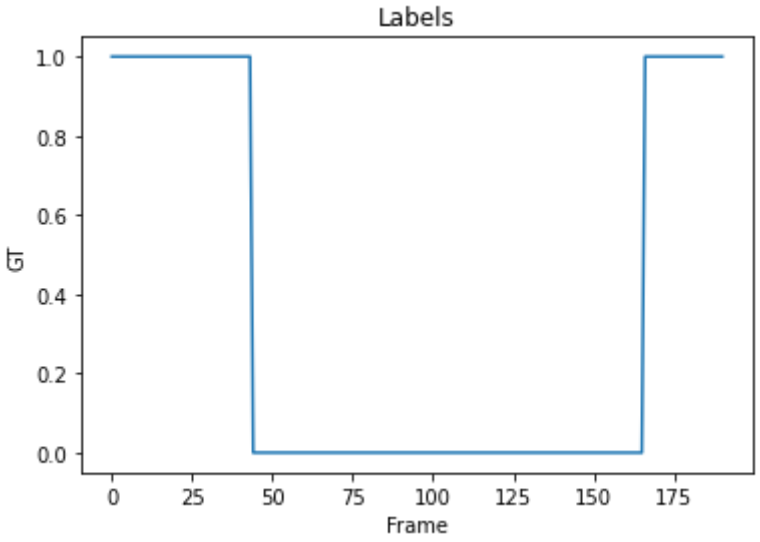



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test024
GT:  24
got model
(200, 227, 227, 1)
got data
```

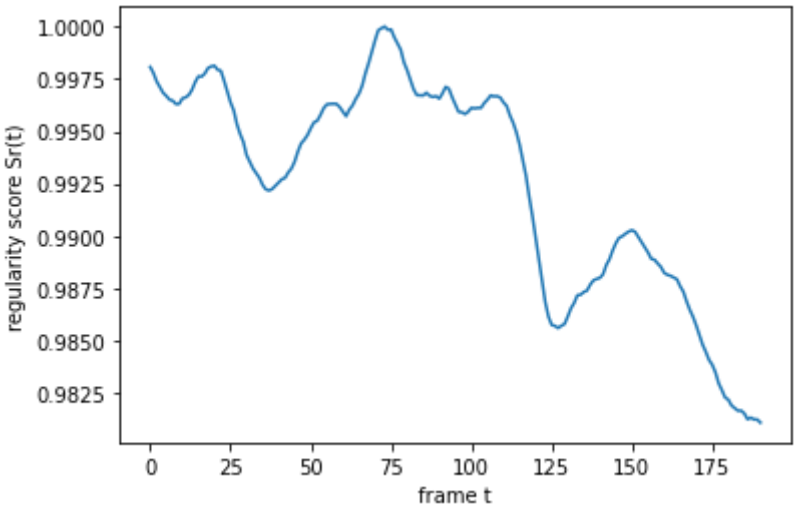



```
AUC:  0.9647184604419102
EER:  0.07377049180327869
EER THRESHOLD:  0.9898562418547154
Optimal threshold value is: 0.9898562418547154
```

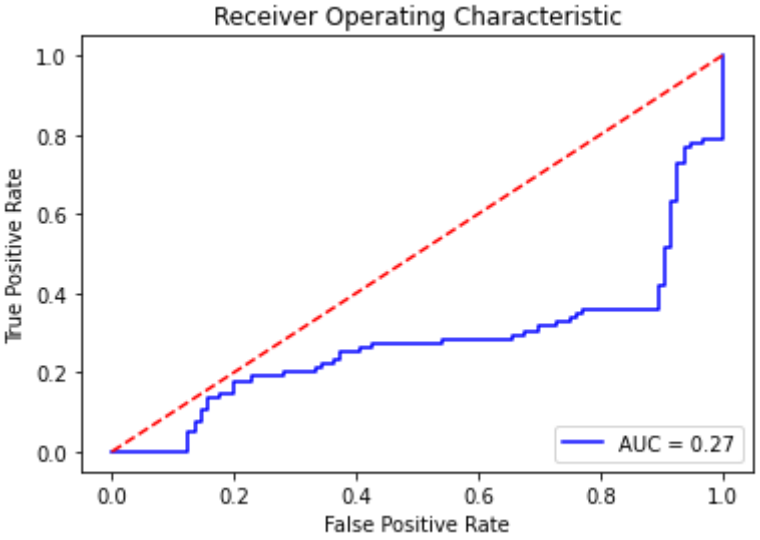


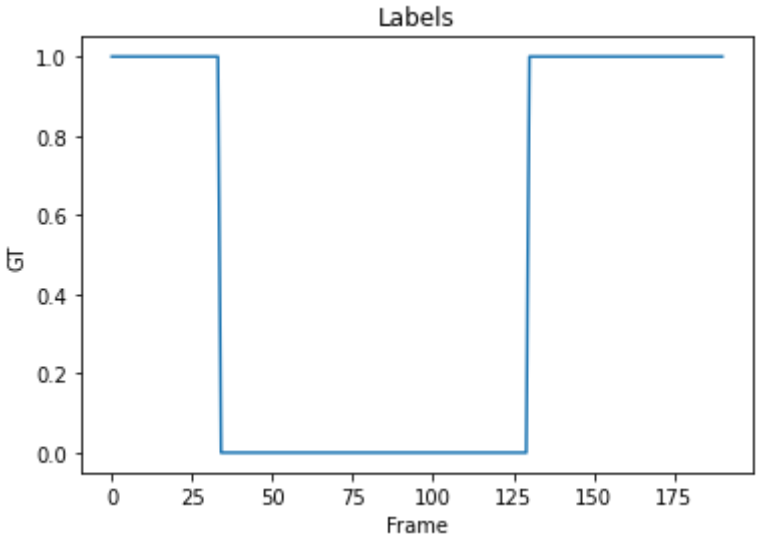



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test025
GT:  25
got model
(200, 227, 227, 1)
got data
```

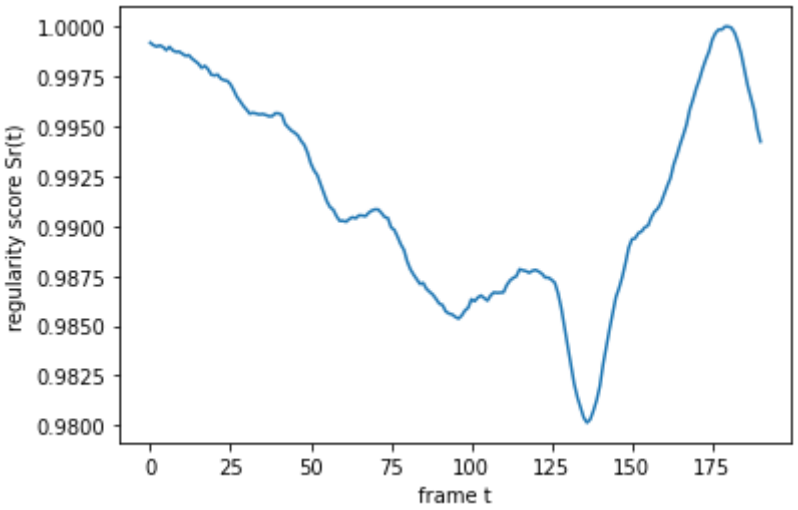



```
AUC:  0.27456140350877195
EER:  0.6979166666666666
EER THRESHOLD:  0.9945993964152043
Optimal threshold value is: 2.0
```

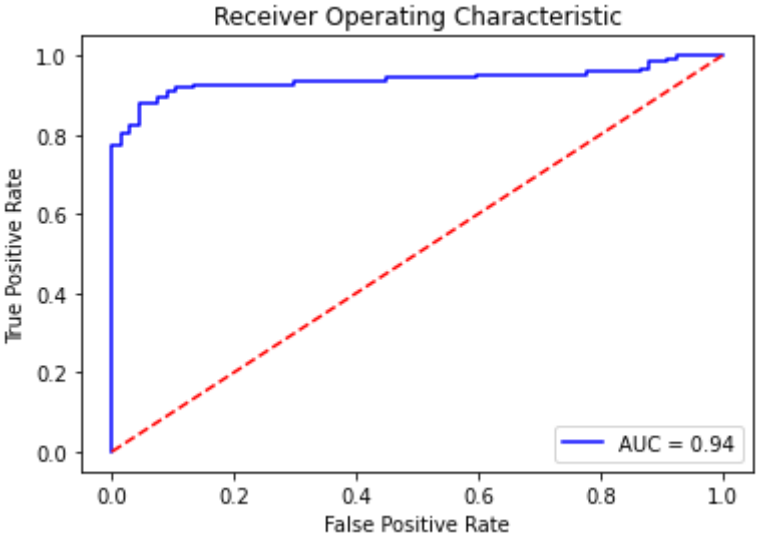


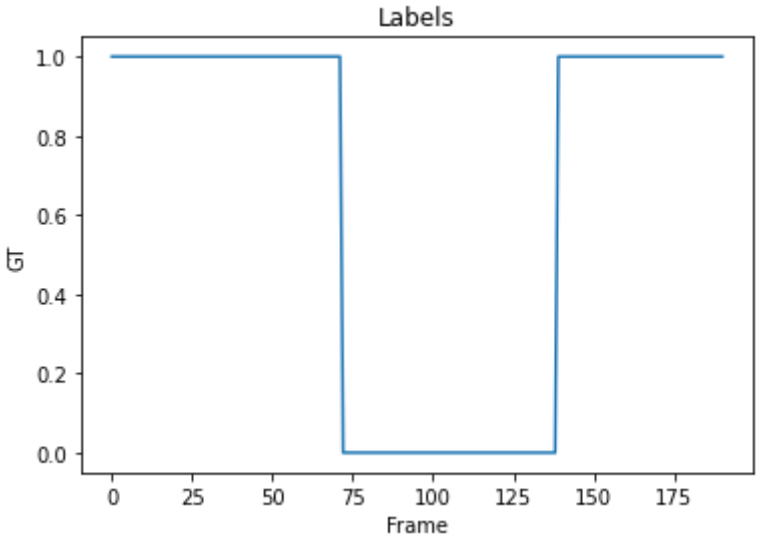



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test026
GT:  26
got model
(200, 227, 227, 1)
got data
```

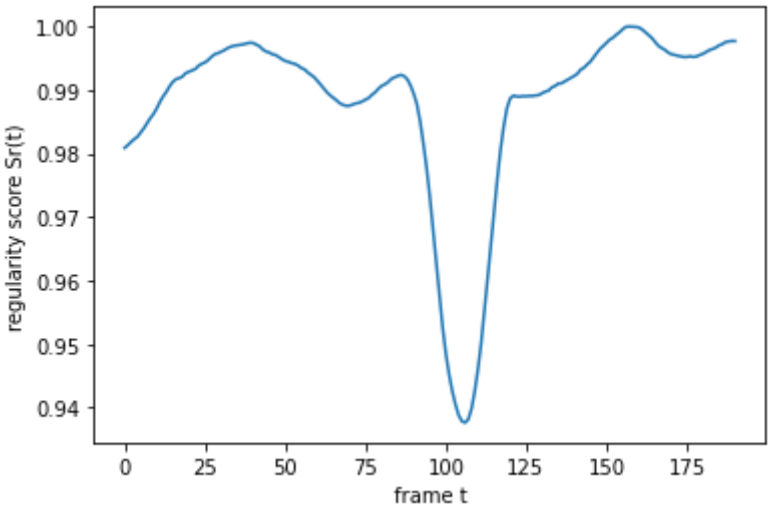



```
AUC:  0.9388541165142033
EER:  0.08955223880597014
EER THRESHOLD:  0.9893380042263302
Optimal threshold value is: 0.9899338476790949
```

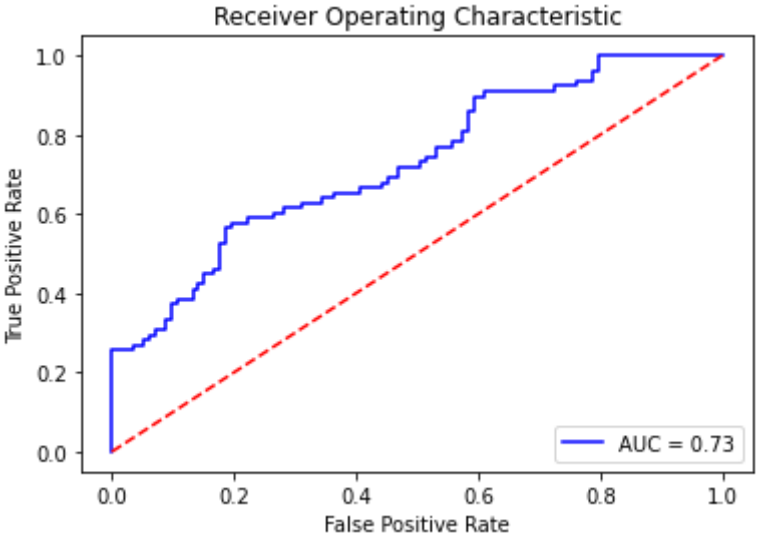


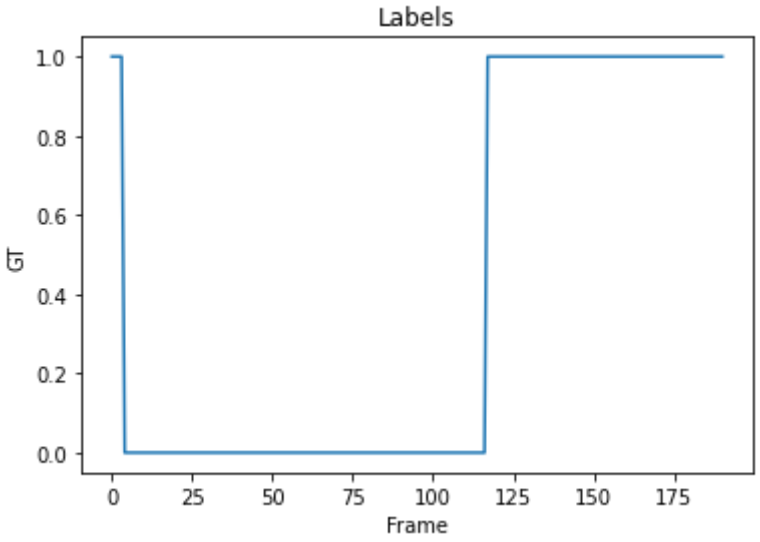



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test027
GT:  27
got model
(200, 227, 227, 1)
got data
```

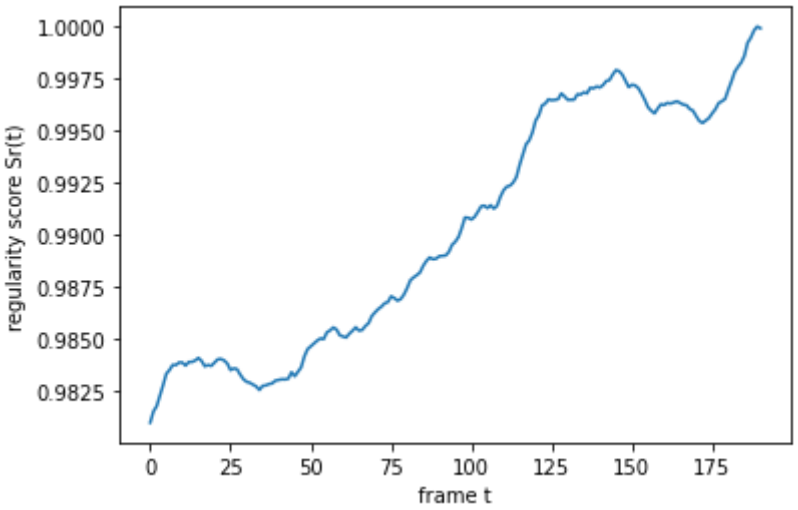



```
AUC:  0.7269117313365102
EER:  0.36283185840707965
EER THRESHOLD:  0.992384752624664
Optimal threshold value is: 0.9951286680888692
```

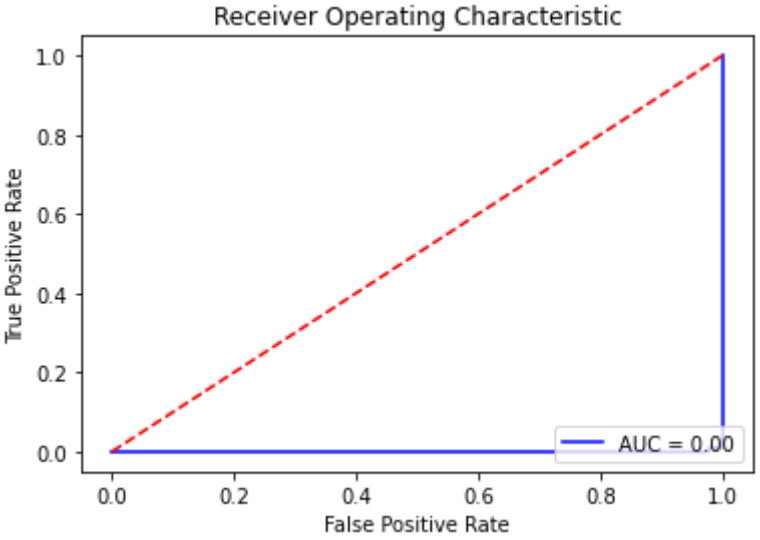


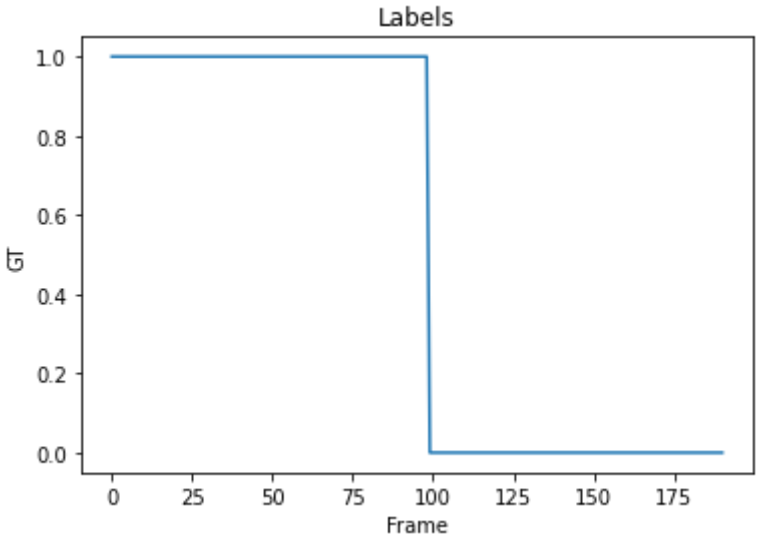



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test028
GT:  28
got model
(200, 227, 227, 1)
got data
```

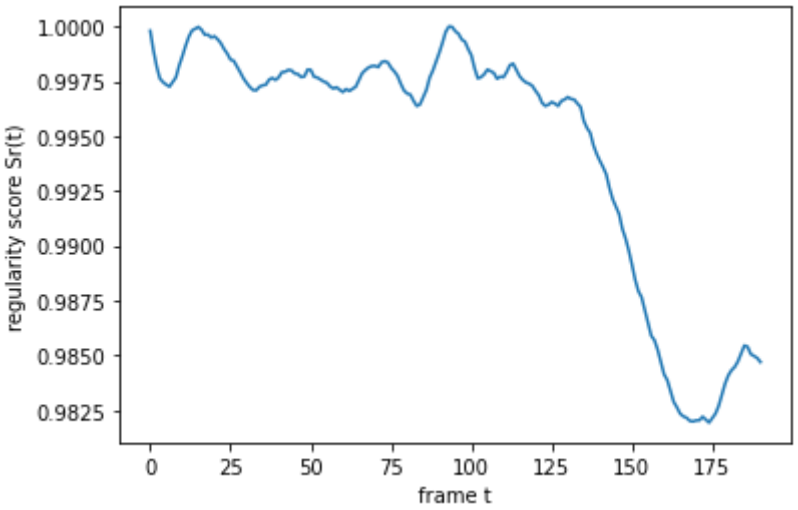



```
AUC:  0.0002195871761089151
EER:  1.0
EER THRESHOLD:  0.9907398043849521
Optimal threshold value is: 2.0
```

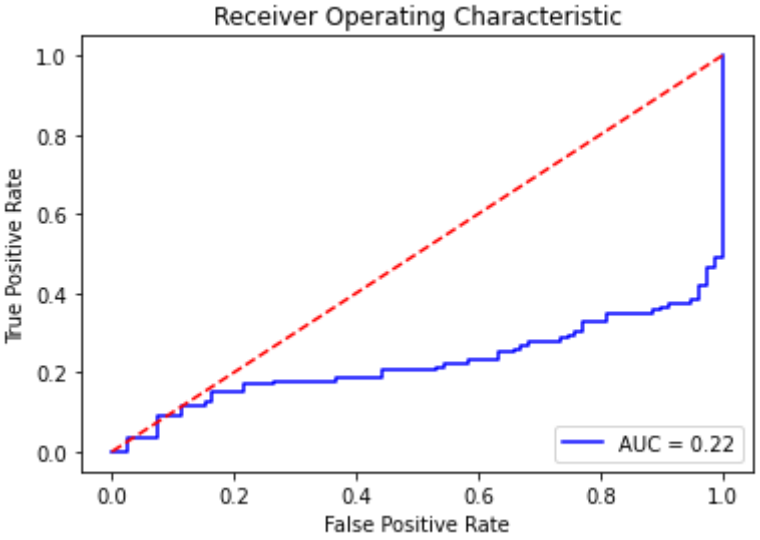


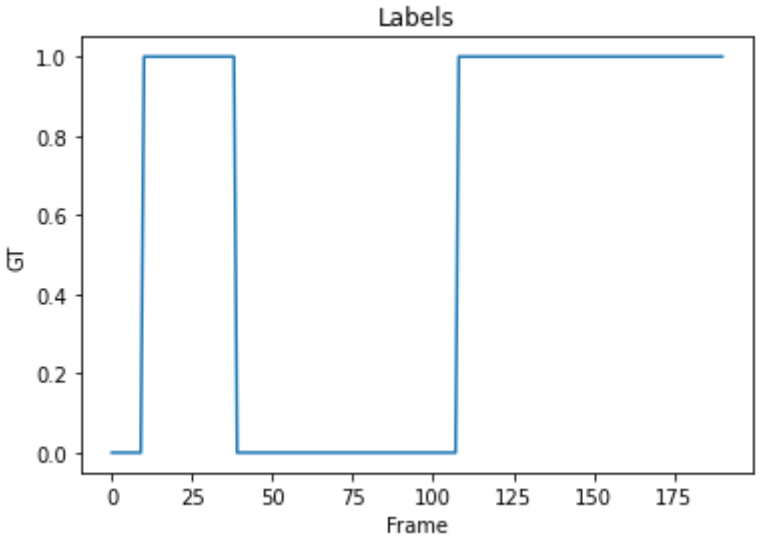



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test029
GT:  29
got model
(200, 227, 227, 1)
got data
```

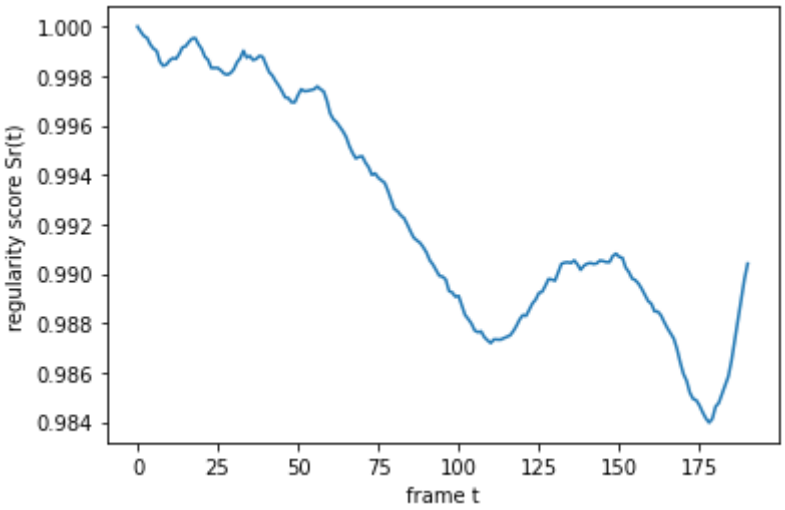



```
AUC:  0.22377938517179022
EER:  0.7341772151898734
EER THRESHOLD:  0.9974633076992875
Optimal threshold value is: 0.9994170917837505
```

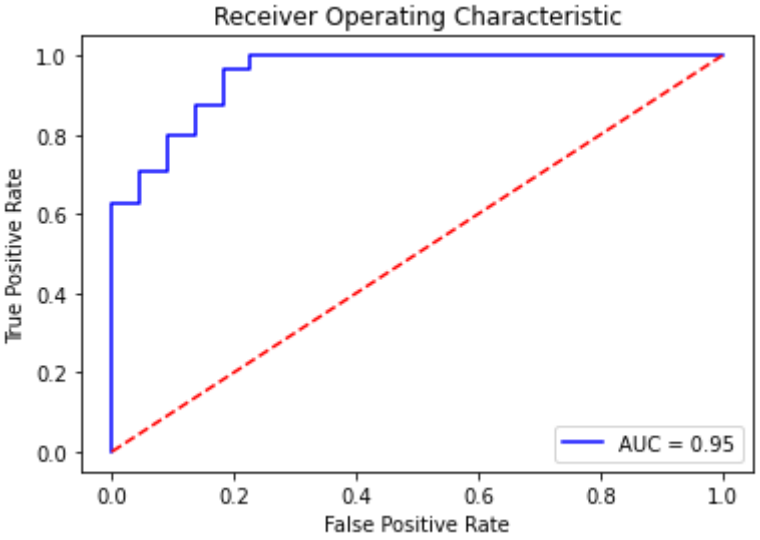


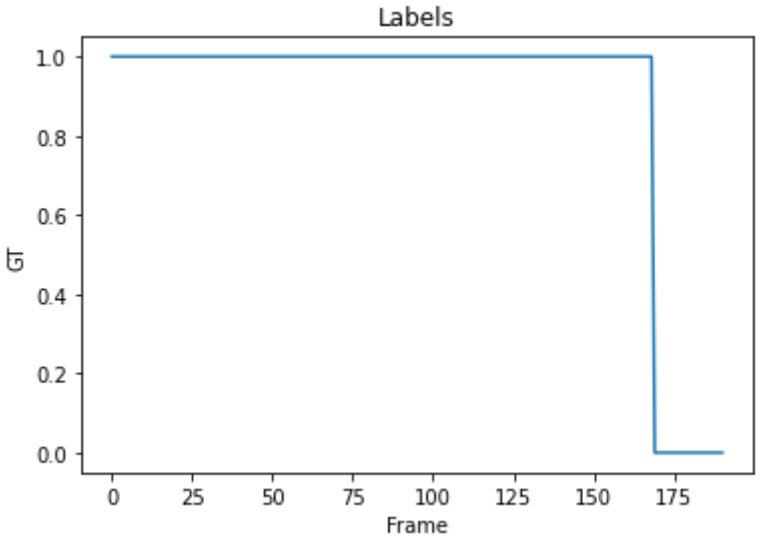



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test030
GT:  30
got model
(200, 227, 227, 1)
got data
```

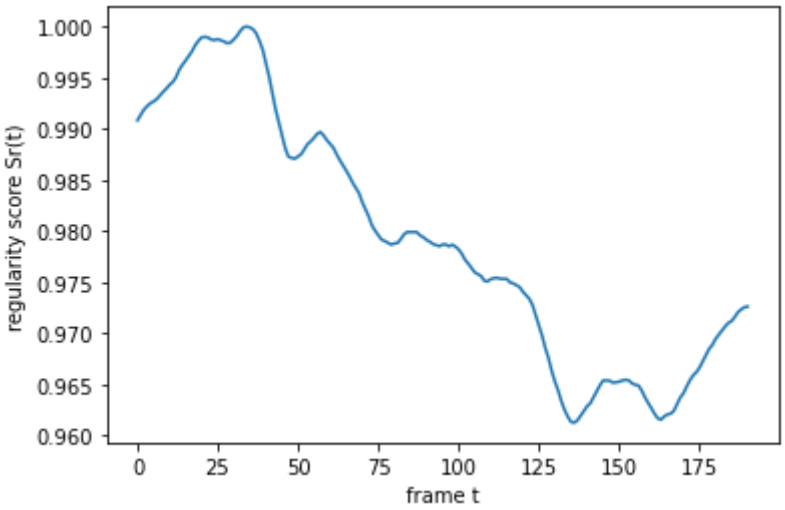



```
AUC:  0.9534696073157611
EER:  0.13636363636363635
EER THRESHOLD:  0.9881689371837791
Optimal threshold value is: 0.9873759000604134
```

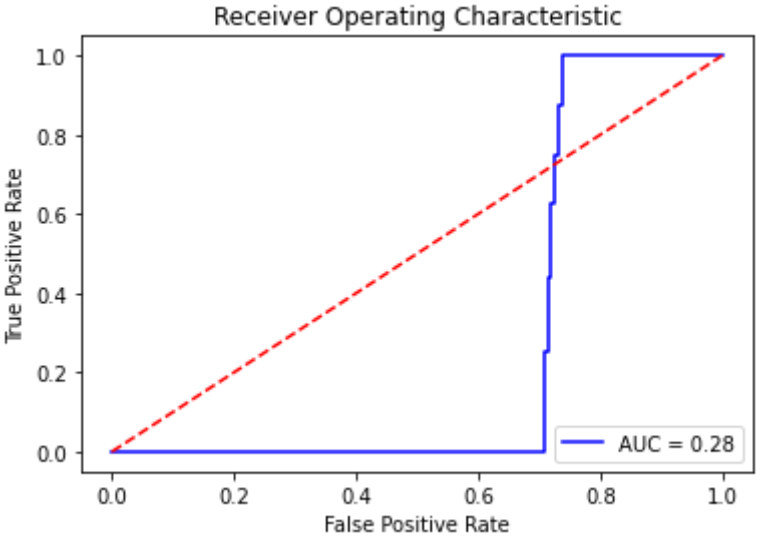


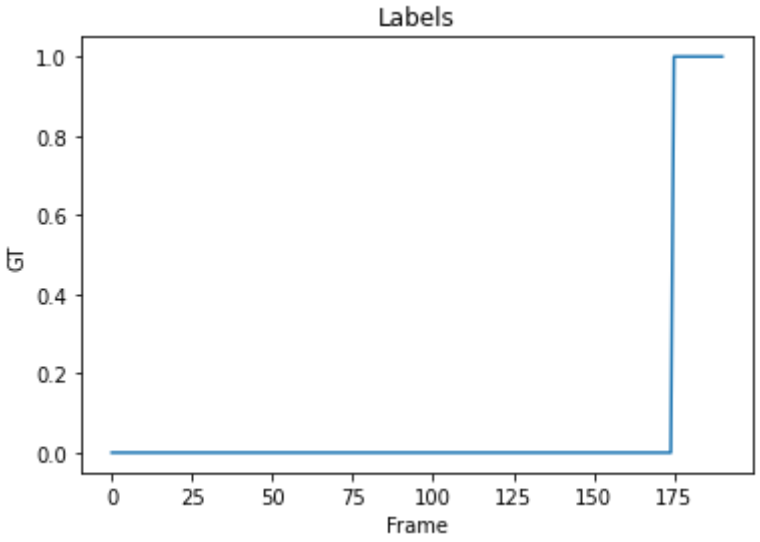



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test031
GT:  31
got model
(200, 227, 227, 1)
got data
```

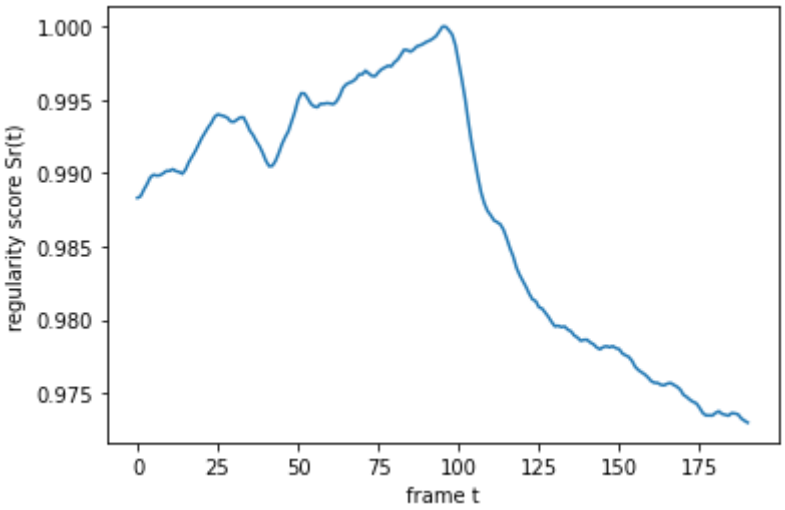



```
AUC:  0.27964285714285714
EER:  0.7142857142857143
EER THRESHOLD:  0.9717757766205203
Optimal threshold value is: 0.9666339601210161
```

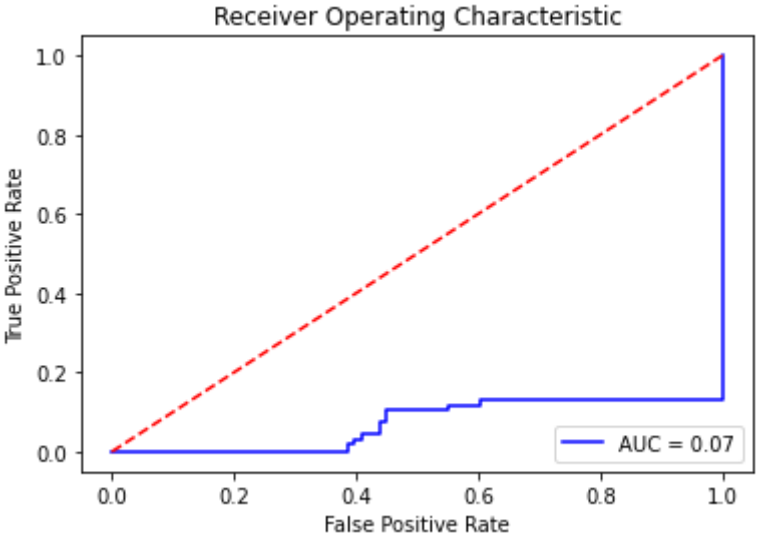


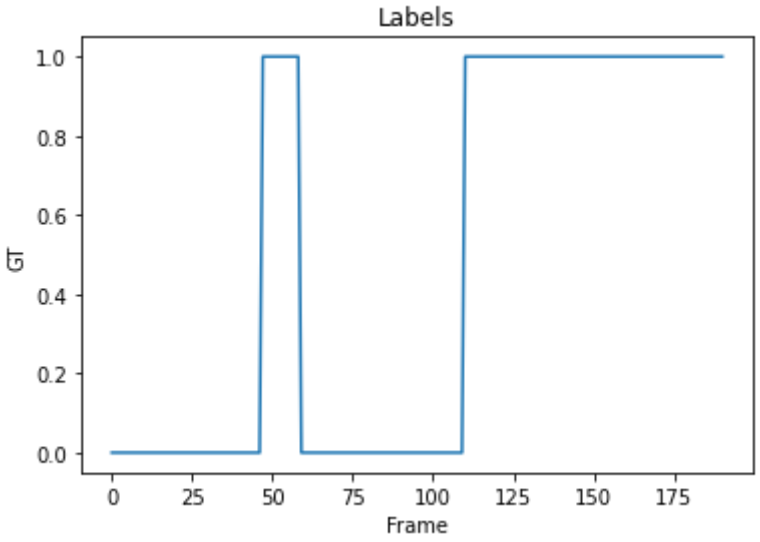



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test032
GT:  32
got model
(200, 227, 227, 1)
got data
```

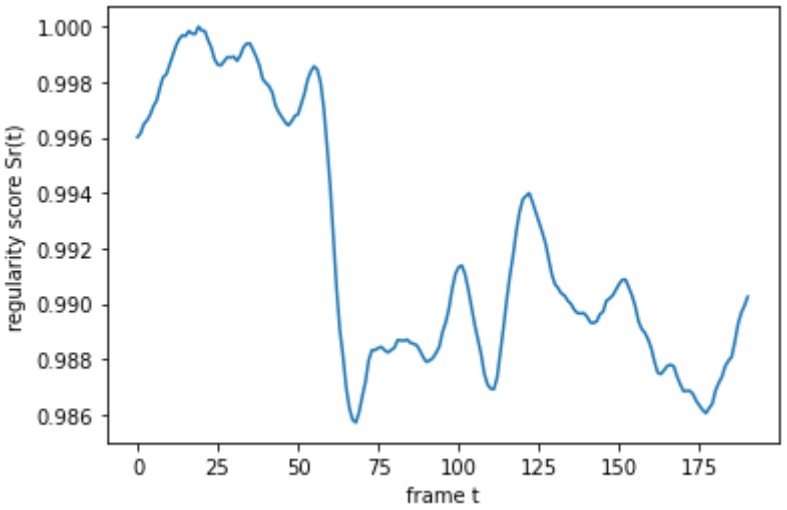



```
AUC:  0.07098968619705948
EER:  1.0
EER THRESHOLD:  0.9874283935391516
Optimal threshold value is: 2.0
```

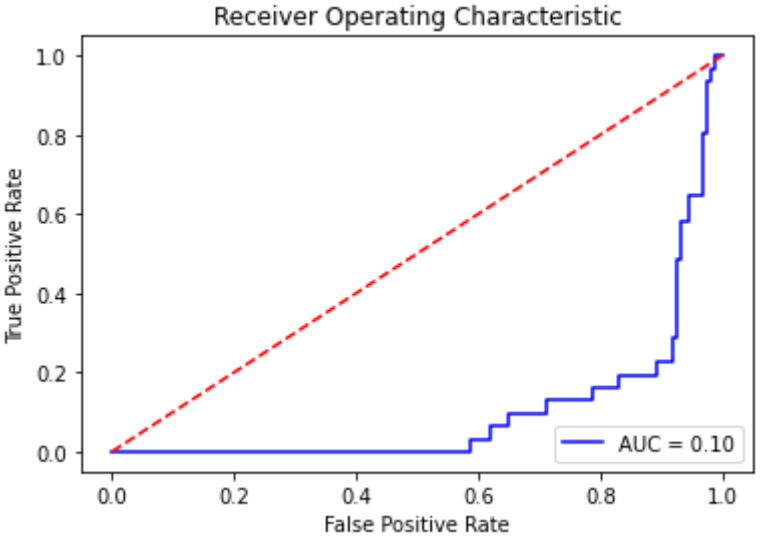


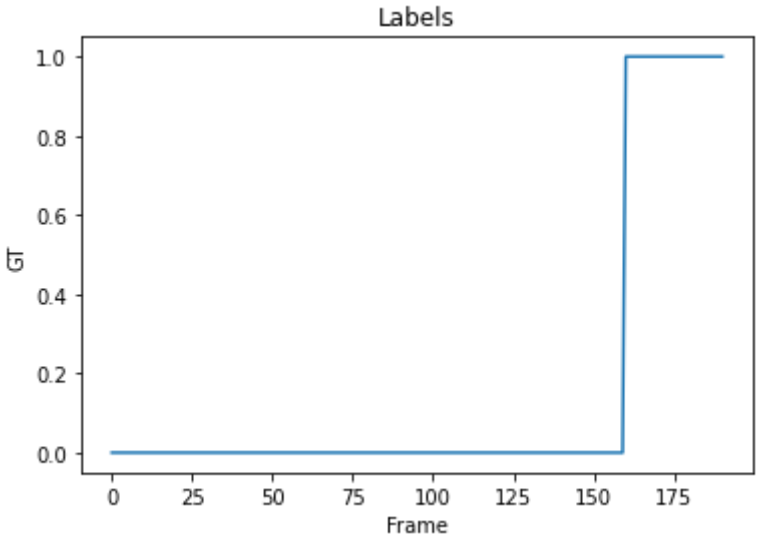



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test033
GT:  33
got model
(200, 227, 227, 1)
got data
```

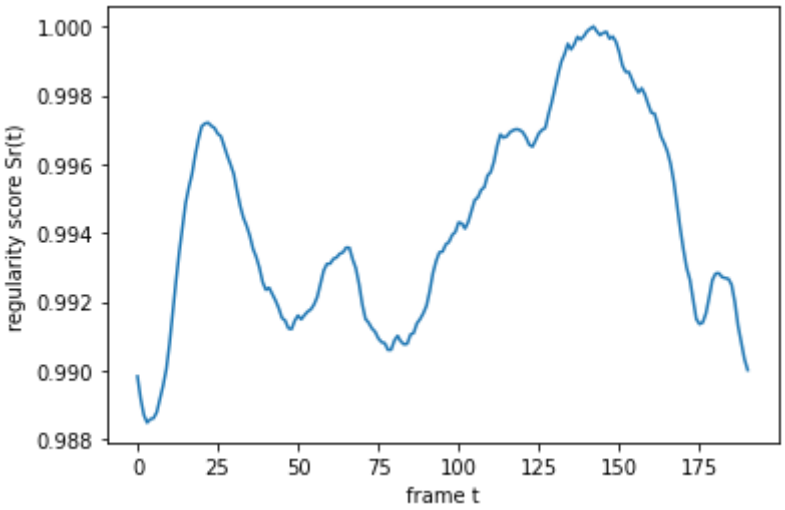



```
AUC:  0.10120967741935483
EER:  0.83125
EER THRESHOLD:  0.9884029267597008
Optimal threshold value is: 0.986052505742211
```

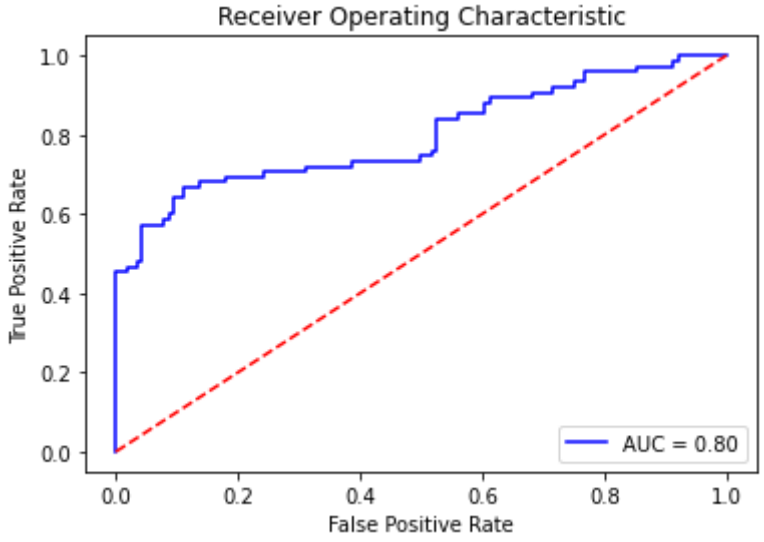


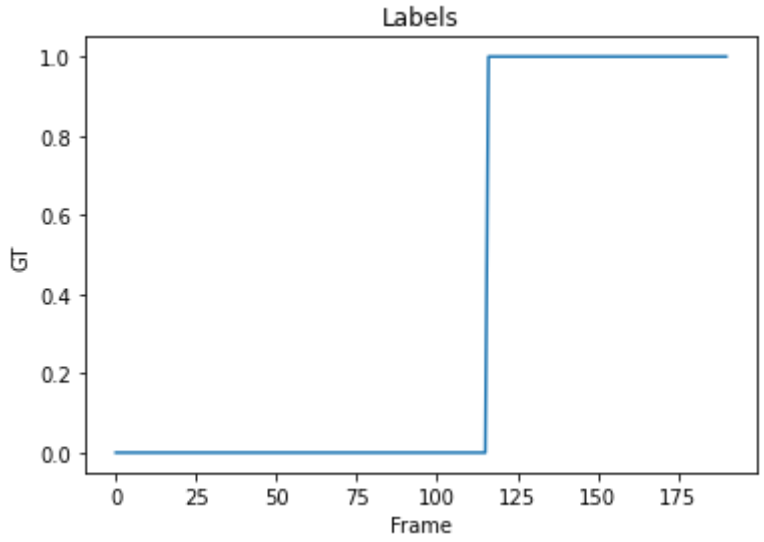



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test034
GT:  34
got model
(200, 227, 227, 1)
got data
```

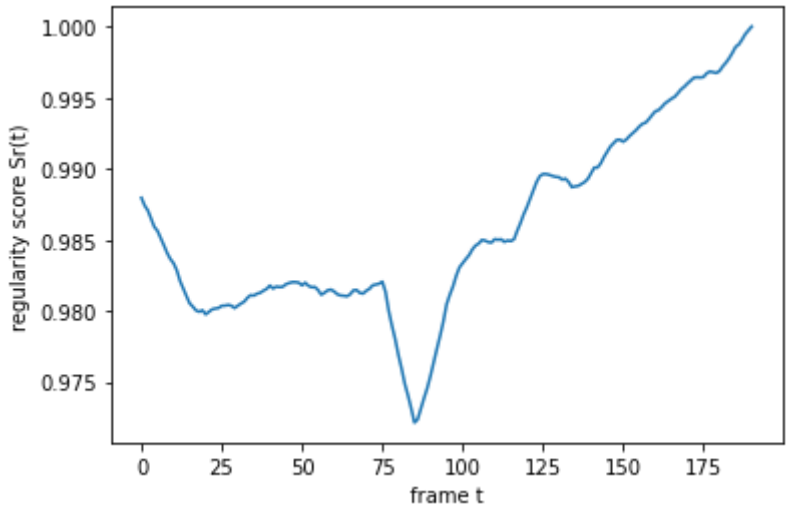



```
AUC:  0.7972413793103448
EER:  0.3103448275862069
EER THRESHOLD:  0.9941563549629839
Optimal threshold value is: 0.9963702457946256
```

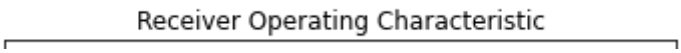





```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test035
GT:  35
got model
(200, 227, 227, 1)
got data
```




```
AUC:  0.16925675675675675
EER:  0.8378378378378378
EER THRESHOLD:  0.9824707642662058
Optimal threshold value is: 0.9780185428466251
```



In [ ]: