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_v10.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()

    #na podstawie oryginalu
    seq.add(TimeDistributed(Conv2D(128, (11, 11), strides=4, padding="valid
    seq.add(LayerNormalization())
    seq.add(TimeDistributed(Conv2D(64, (5, 5), strides=2, padding="valid"))
    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(128, (5, 5), strides=2, padding
    seq.add(LayerNormalization())
    seq.add(TimeDistributed(Conv2DTranspose(1, (11, 11), strides=4, padding
    seq.add(LayerNormalization())
    #seq.add(TimeDistributed(Conv2D(1, (11, 11), activation="sigmoid", pad

    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="valid
    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_sequen
    seq.add(LayerNormalization())
    seq.add(ConvLSTM2D(32, (3, 3), strides=1, padding="same", return_sequen
    seq.add(LayerNormalization())
    seq.add(ConvLSTM2D(64, (3, 3), strides=1, padding="same", return_sequen
    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
```

```
2021-11-12 17:45:32.293968: 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
2021-11-12 17:45:32.294034: 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
```

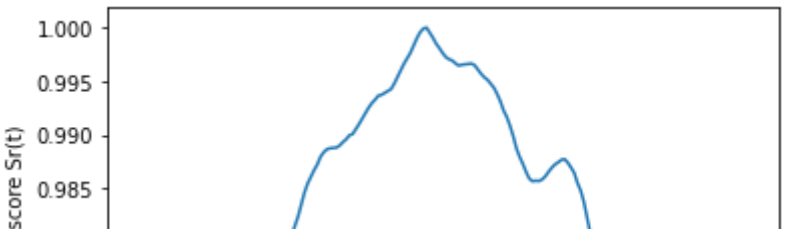
In [9]:

```
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=Config.BA
    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=False)
```

```
2021-11-12 17:45:34.222478: 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
2021-11-12 17:45:34.222520: W tensorflow/stream_executor/cuda/cuda_driver.c
c:269] failed call to cuInit: UNKNOWN ERROR (303)
2021-11-12 17:45:34.222541: 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
2021-11-12 17:45:34.222808: 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.
got model
(200, 227, 227, 1)
got data
```

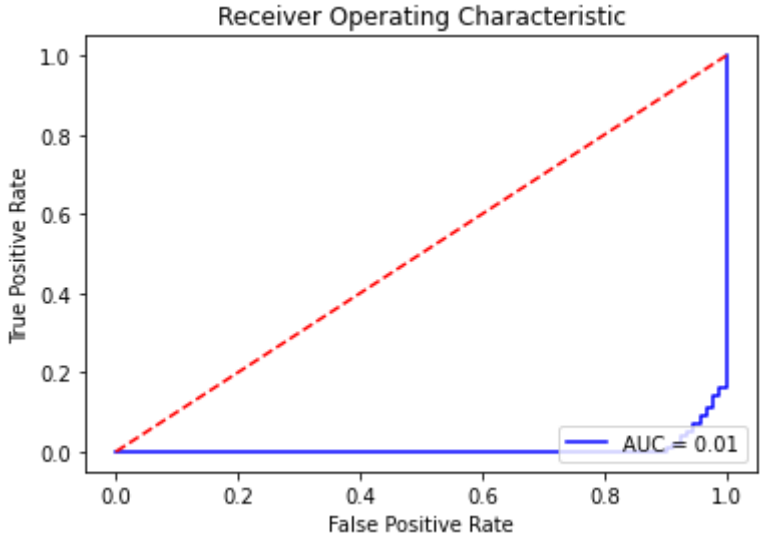


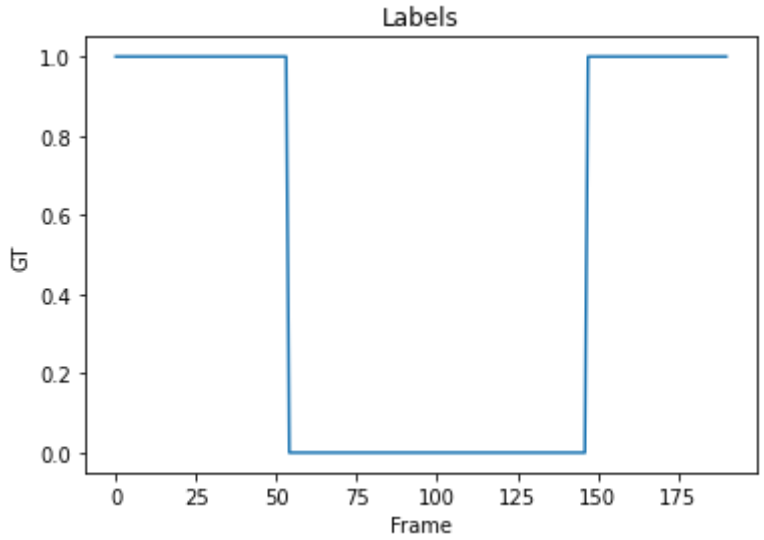
In [11]:

```
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:  0.007570770243581302
```

```
EER:  0.946236559139785
EER THRESHOLD:  0.9808119557112213
Optimal threshold value is: 2.0
```



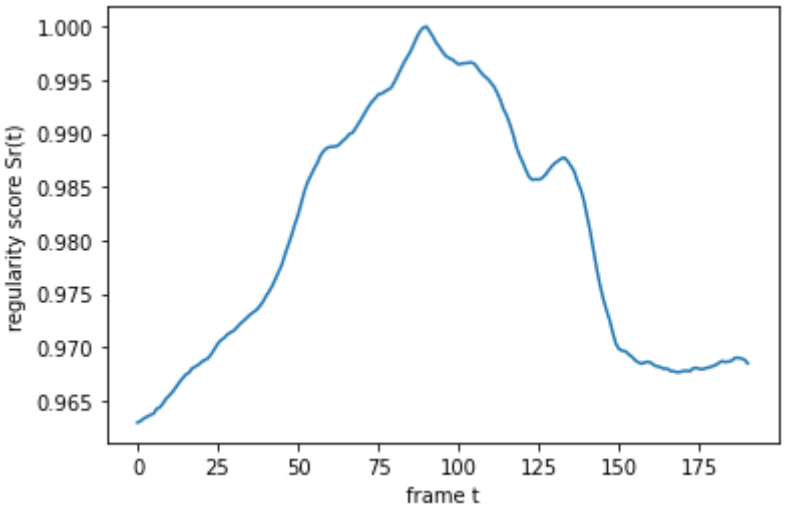


Out[11]: (0.007570770243581302, 0.946236559139785)

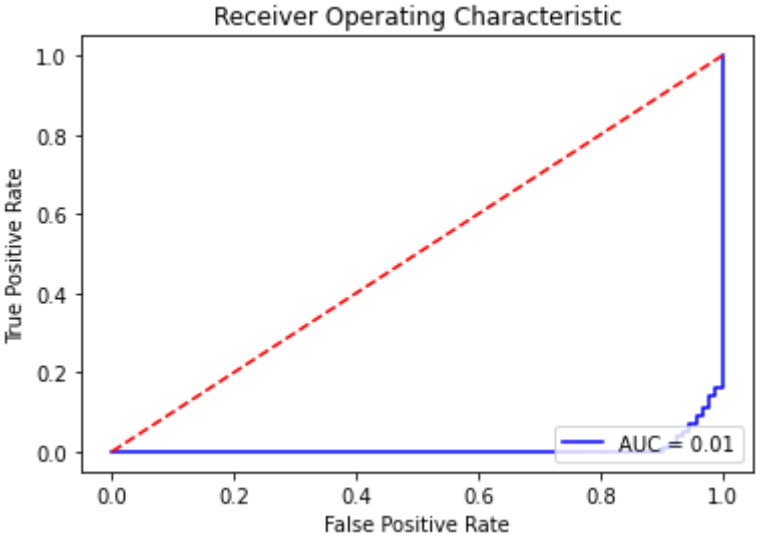
In [12]:

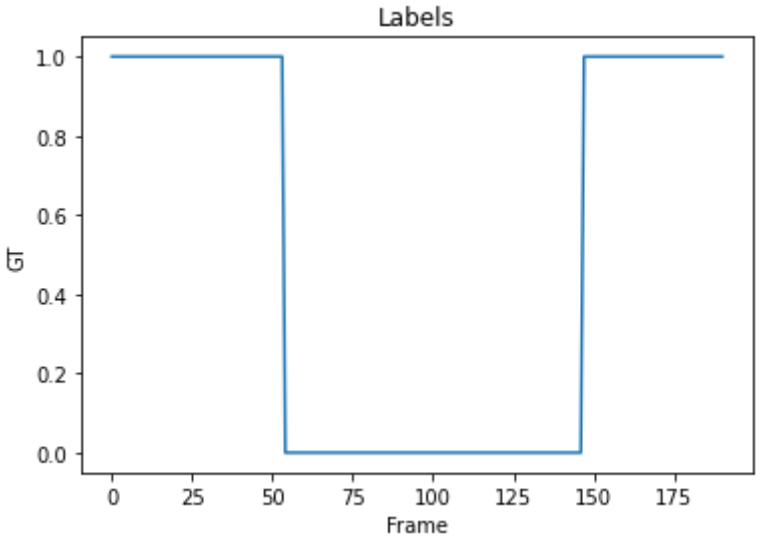
```
from os import listdir
from os.path import isfile, join, isdir


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
```

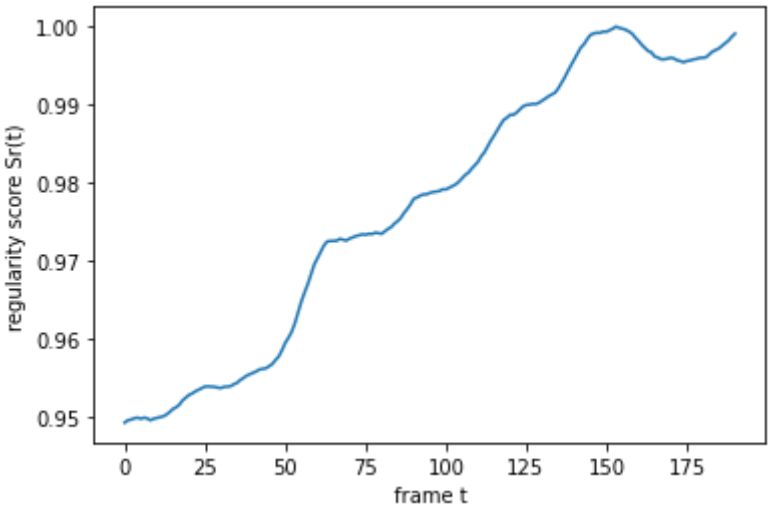



```
AUC:  0.007570770243581302
EER:  0.946236559139785
EER THRESHOLD:  0.9808119557112213
Optimal threshold value is: 2.0
```

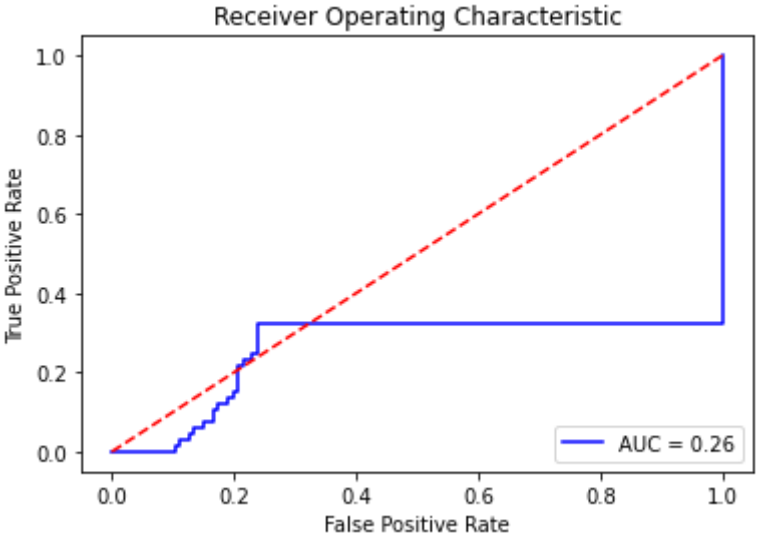


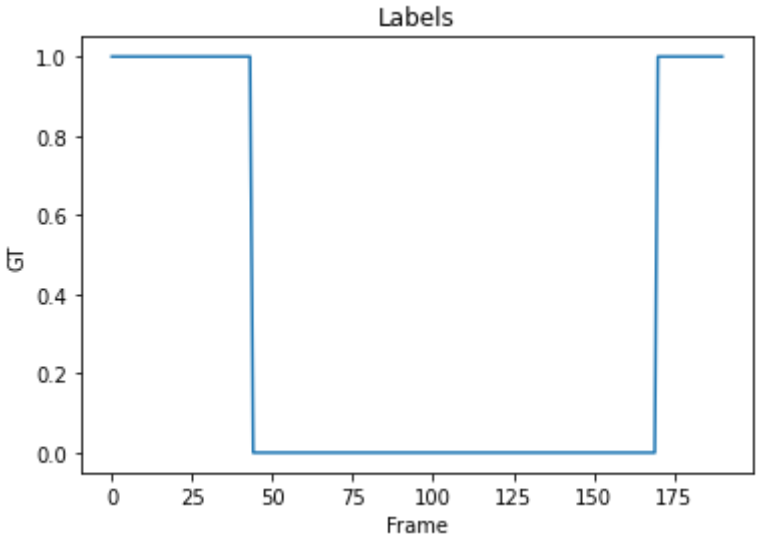



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test002
GT:  2
got model
(200, 227, 227, 1)
got data
```

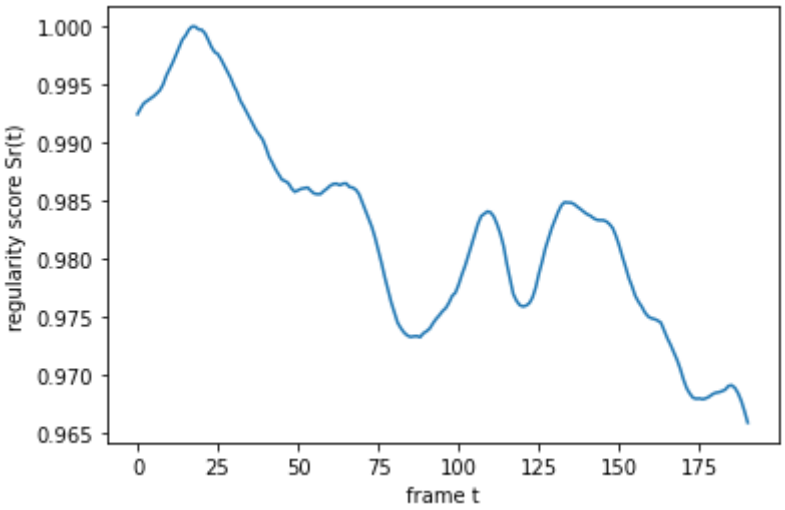



```
AUC:  0.2617826617826618
EER:  1.0
EER THRESHOLD:  0.9562331891227641
Optimal threshold value is: 0.9954184295504312
```

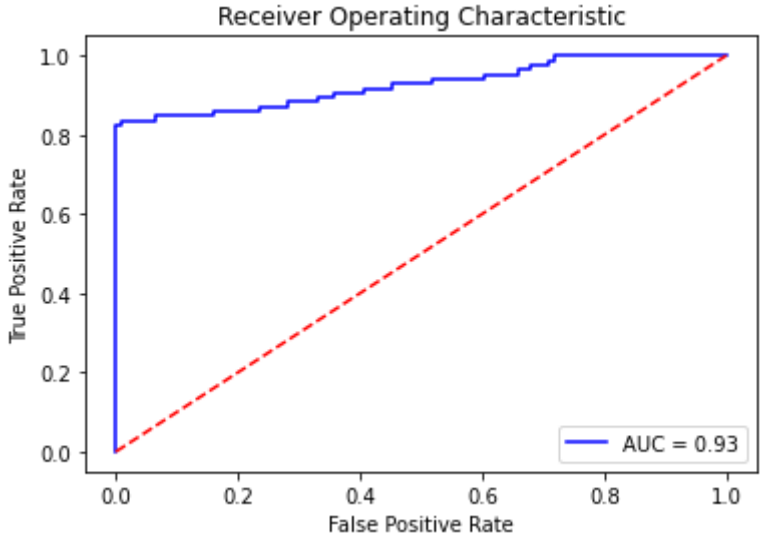


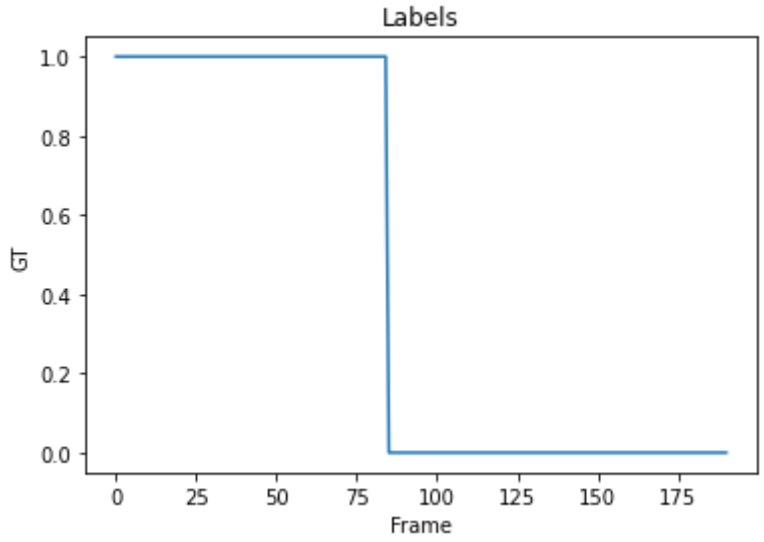



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test003
GT:  3
got model
(200, 227, 227, 1)
got data
```

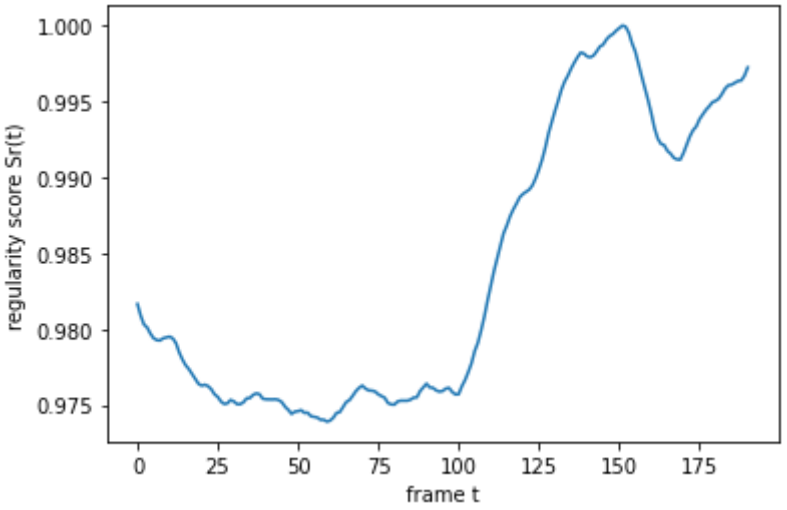



```
AUC:  0.9271920088790233
EER:  0.16037735849056603
EER THRESHOLD:  0.9834851597515305
Optimal threshold value is: 0.9848602326754593
```

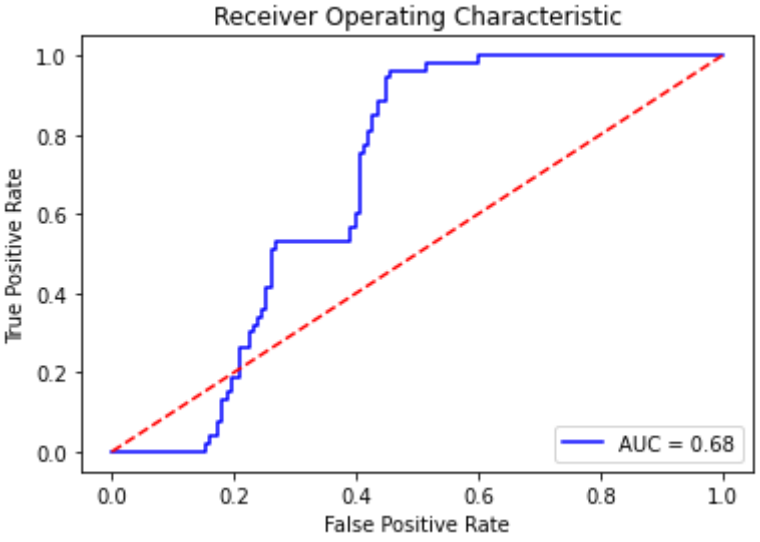


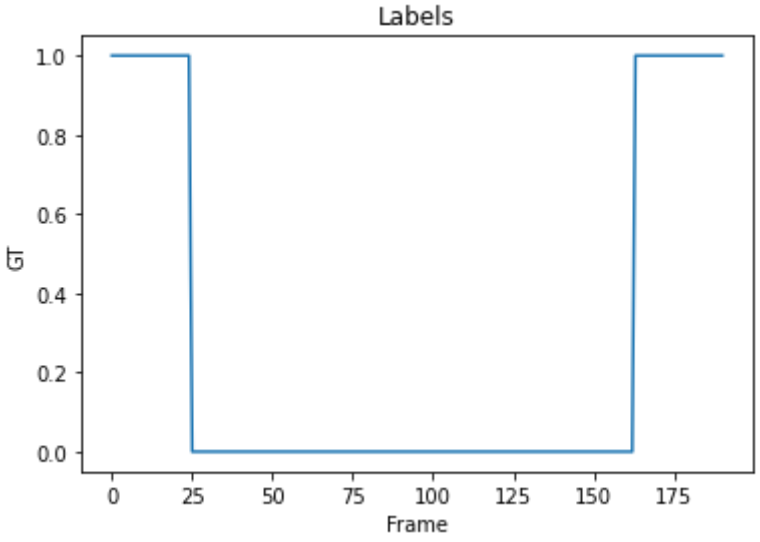



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test004
GT:  4
got model
(200, 227, 227, 1)
got data
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_predi
ct_function.<locals>.predict_function at 0x7f24ac64d280> triggered tf.funct
ion retracing. Tracing is expensive and the excessive number of tracings co
uld be due to (1) creating @tf.function repeatedly in a loop, (2) passing t
ensors with different shapes, (3) passing Python objects instead of tensor
s. For (1), please define your @tf.function outside of the loop. For (2), @
tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to http
s://www.tensorflow.org/guide/function#controlling_retracing and https://ww
w.tensorflow.org/api_docs/python/tf/function for  more details.
```



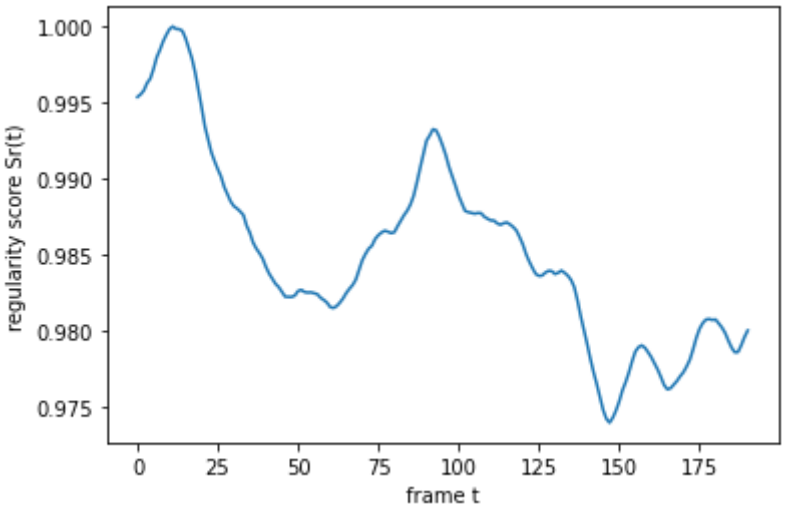

```
AUC:  0.6821164889253486
EER:  0.39855072463768115
EER THRESHOLD:  0.9801105097772063
Optimal threshold value is: 0.9762370793877078
```



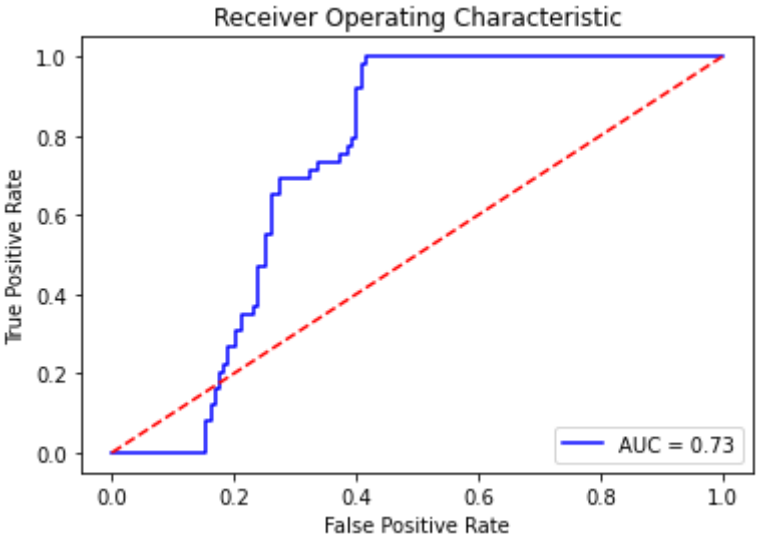


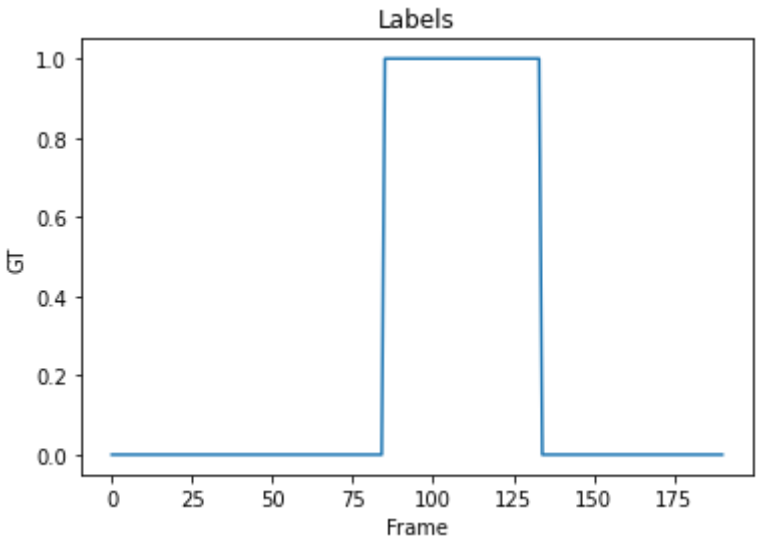

```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test005
GT:  5
got model
(200, 227, 227, 1)
got data
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_predi
ct_function.<locals>.predict_function at 0x7f238ab889d0> triggered tf.funct
```

```
ion retracing. Tracing is expensive and the excessive number of tracings co
uld be due to (1) creating @tf.function repeatedly in a loop, (2) passing t
ensors with different shapes, (3) passing Python objects instead of tensor
s. For (1), please define your @tf.function outside of the loop. For (2), @
tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to http
s://www.tensorflow.org/guide/function#controlling_retracing and https://ww
```

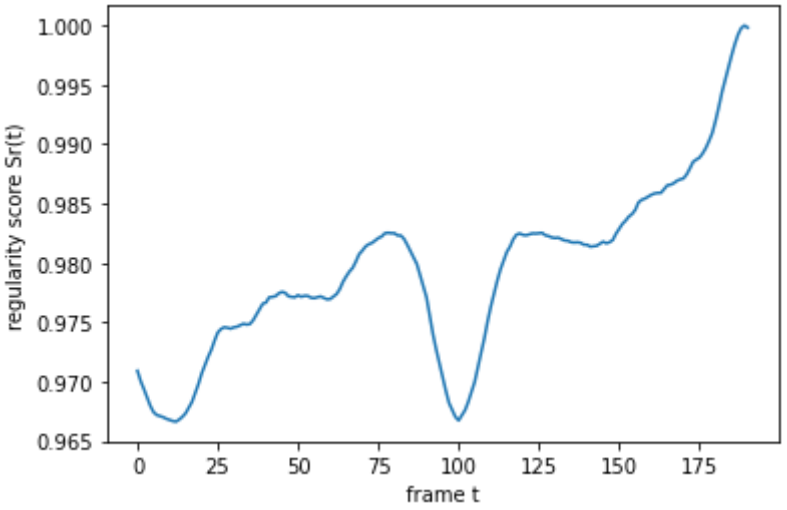



```
AUC:  0.73081345214142
EER:  0.323943661971831
EER THRESHOLD:  0.9863087772037304
Optimal threshold value is: 0.9836389022322012
```

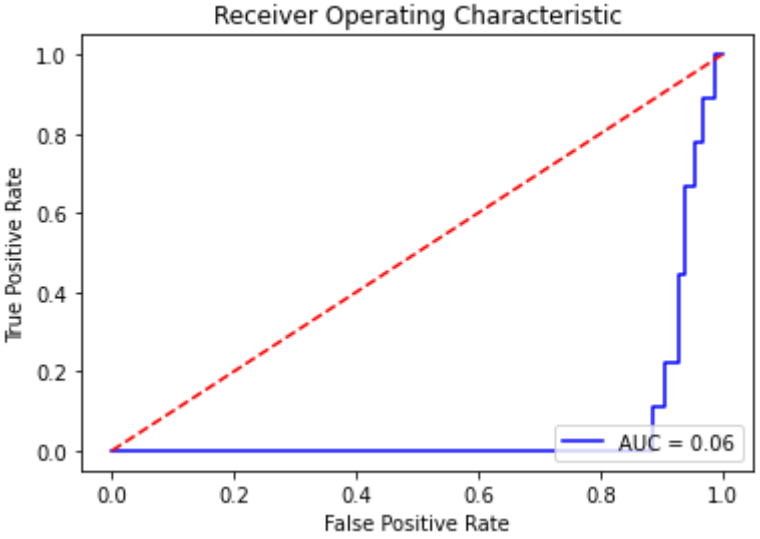


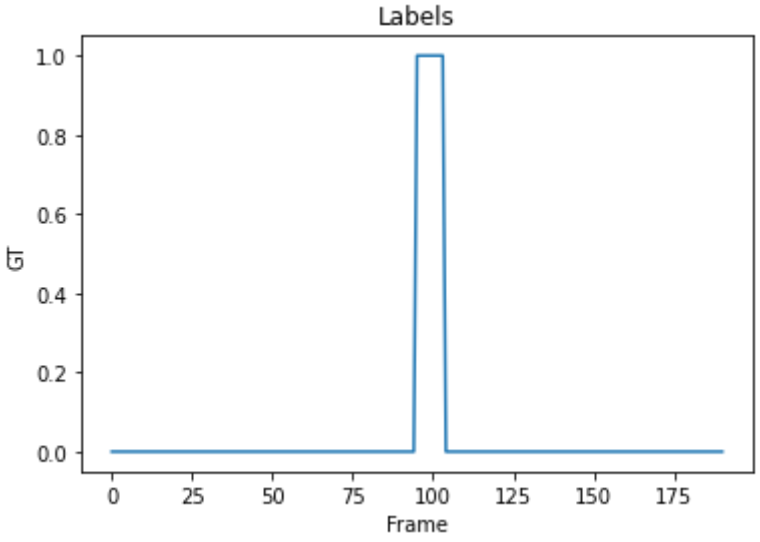


```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test006
GT:  6
got model
(200, 227, 227, 1)
got data
```



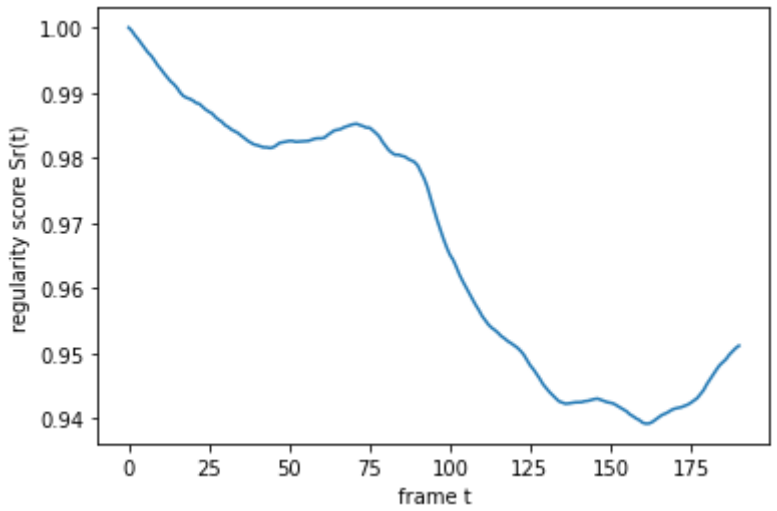

```
AUC:  0.06227106227106226
EER:  0.8846153846153846
EER THRESHOLD:  0.9703860868690168
Optimal threshold value is: 0.9667408034996895
```



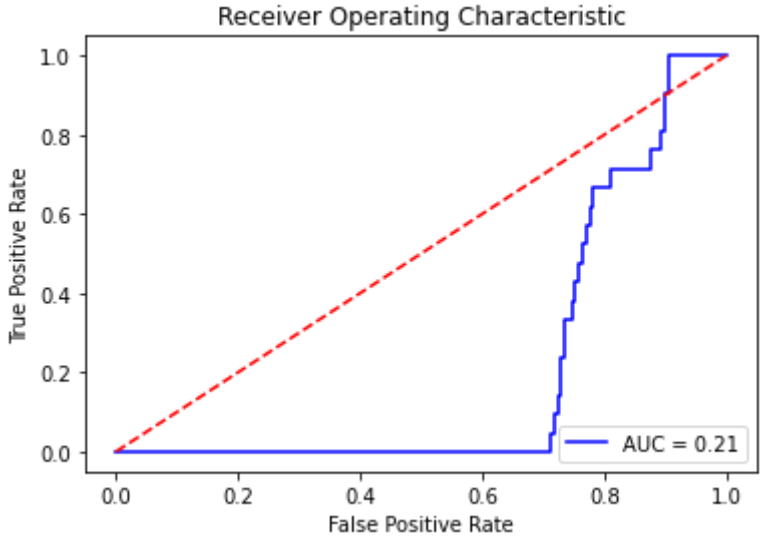


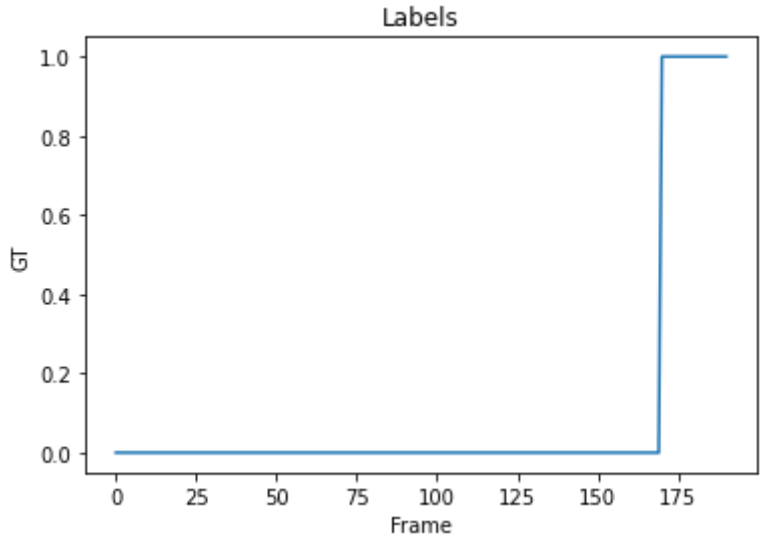

```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test007
GT:  7
```

```
got model
(200, 227, 227, 1)
```

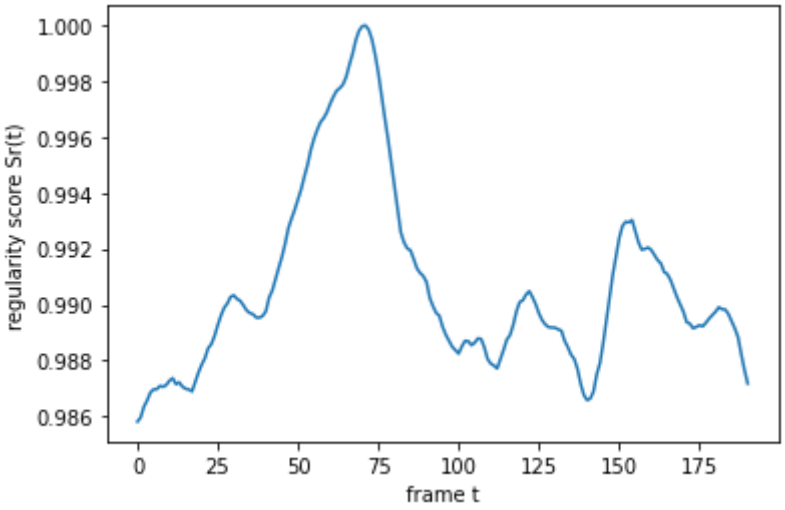



```
AUC:  0.20812324929971987
EER:  0.7352941176470589
EER THRESHOLD:  0.9490413750904821
Optimal threshold value is: 0.9415472785134379
```

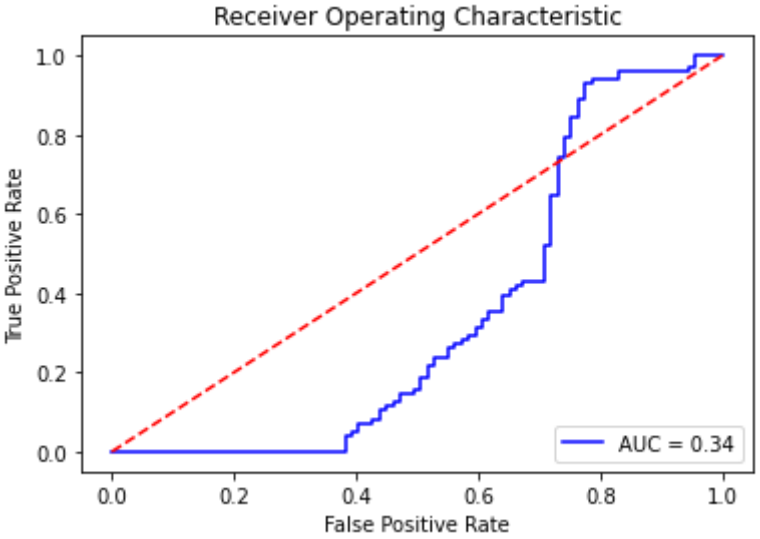


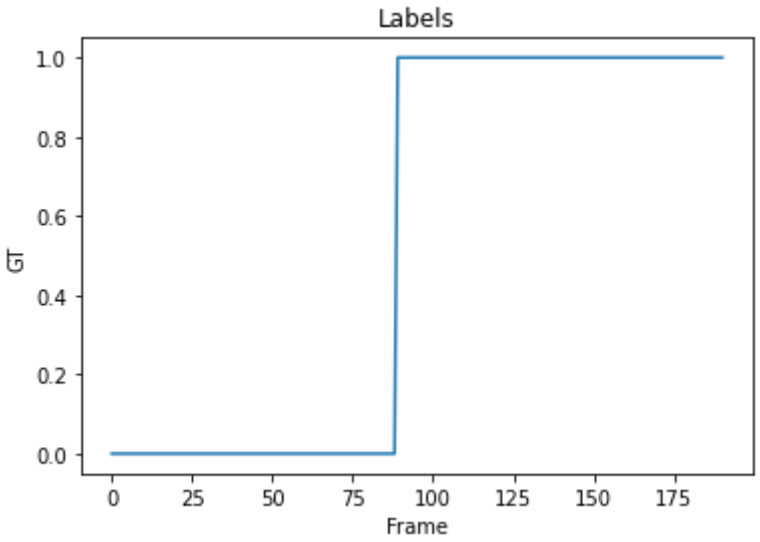



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test008
GT:  8
got model
(200, 227, 227, 1)
got data
```

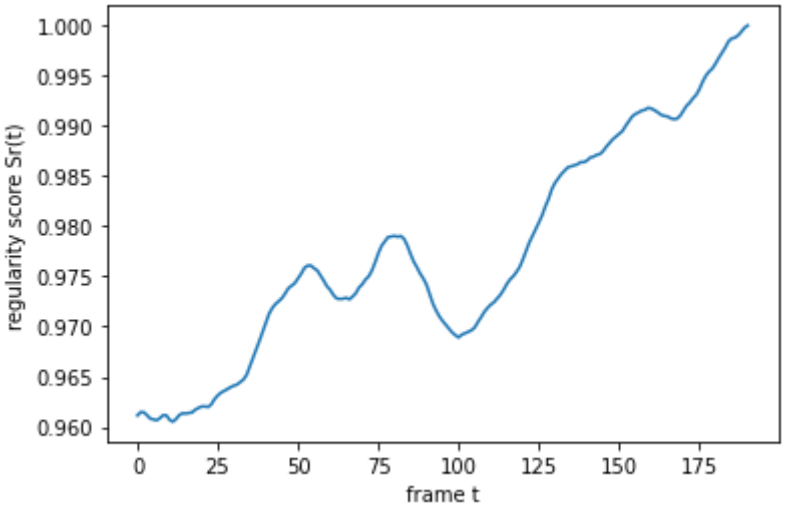



```
AUC:  0.34478960123375196
EER:  0.6404494382022472
EER THRESHOLD:  0.9897635229961608
Optimal threshold value is: 0.9876573774961486
```

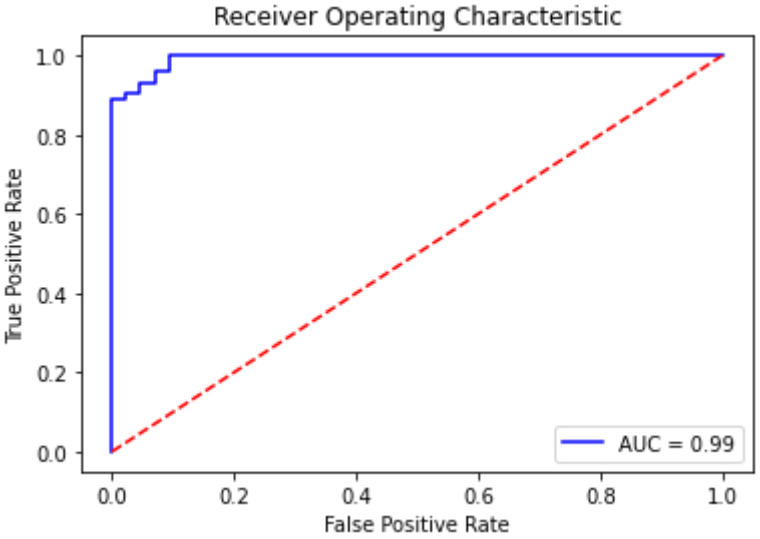


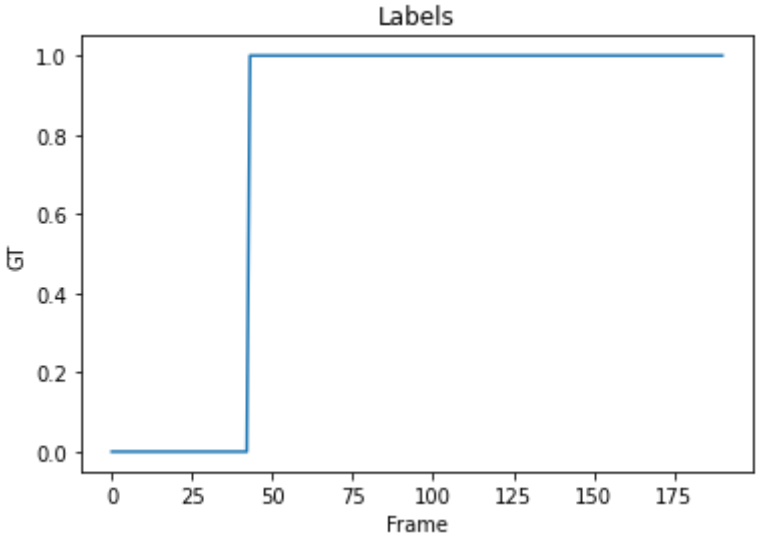



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test009
GT:  9
got model
(200, 227, 227, 1)
got data
```

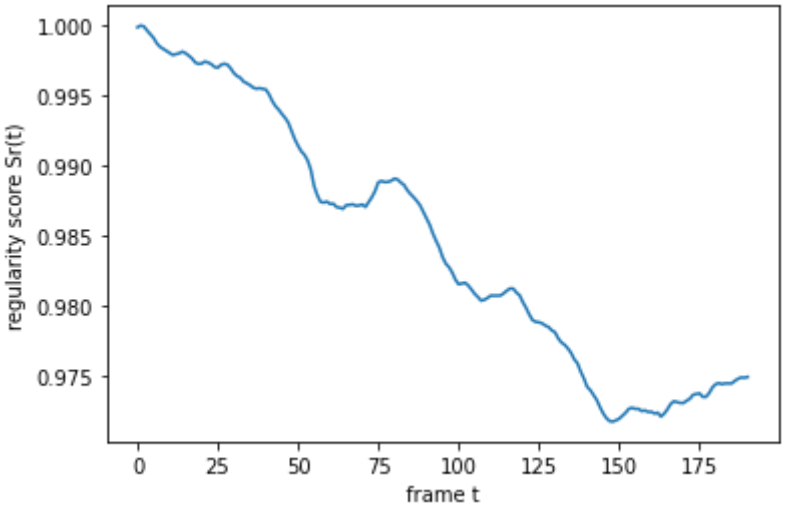



```
AUC:  0.992771841609051
EER:  0.06976744186046512
EER THRESHOLD:  0.970401626639865
Optimal threshold value is: 0.9689135440686959
```

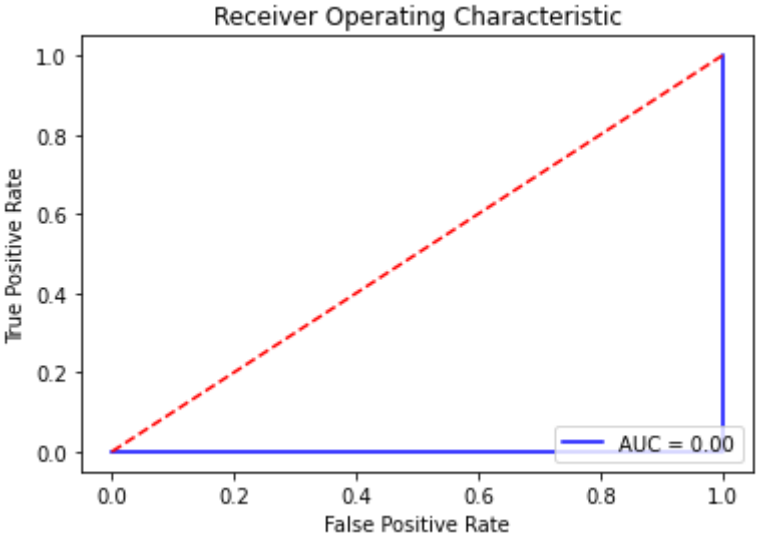


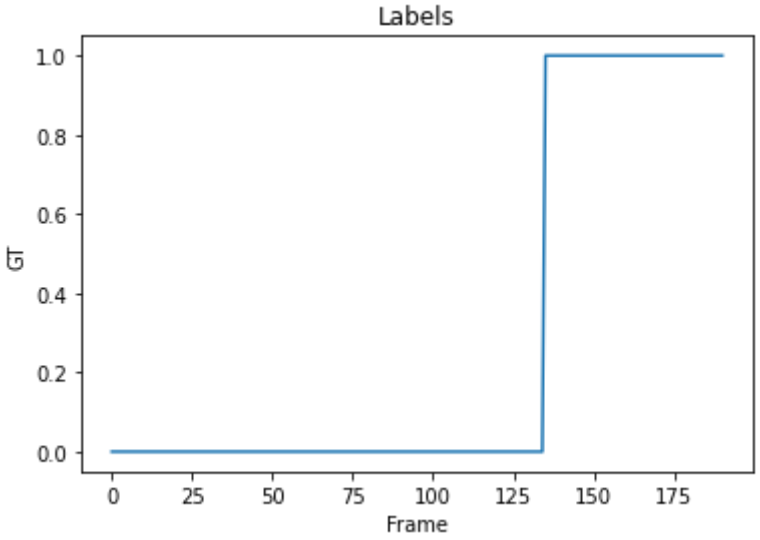



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test010
GT:  10
got model
(200, 227, 227, 1)
got data
```

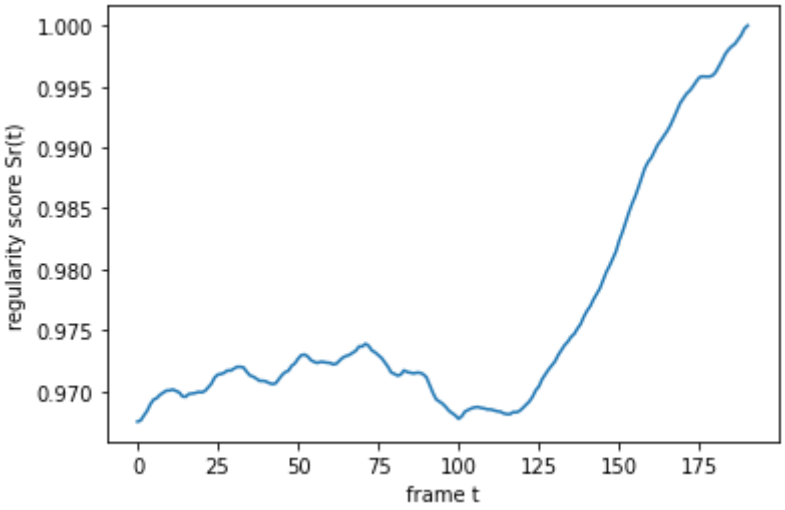



```
AUC:  0.0
EER:  1.0
EER THRESHOLD:  0.9770066778987285
Optimal threshold value is: 2.0
```

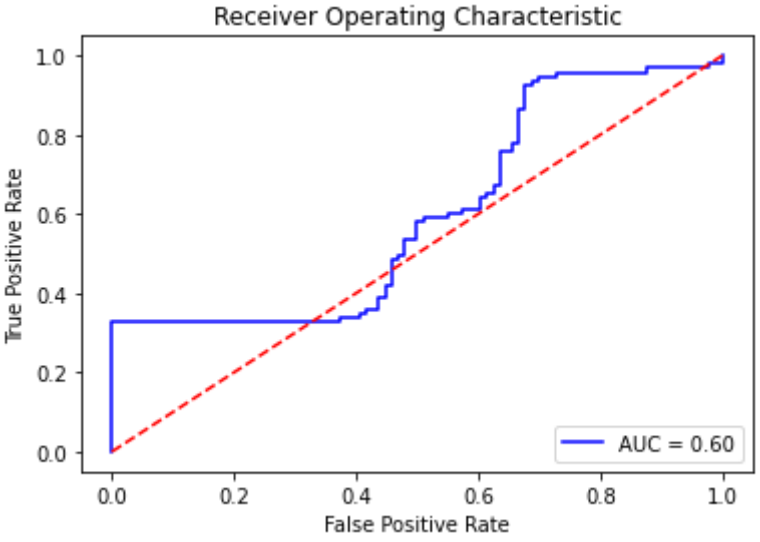


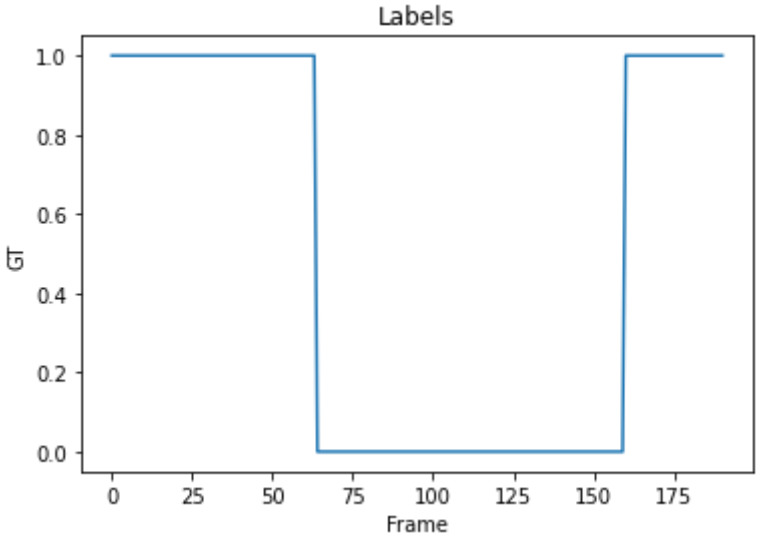



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test011
GT:  11
got model
(200, 227, 227, 1)
got data
```

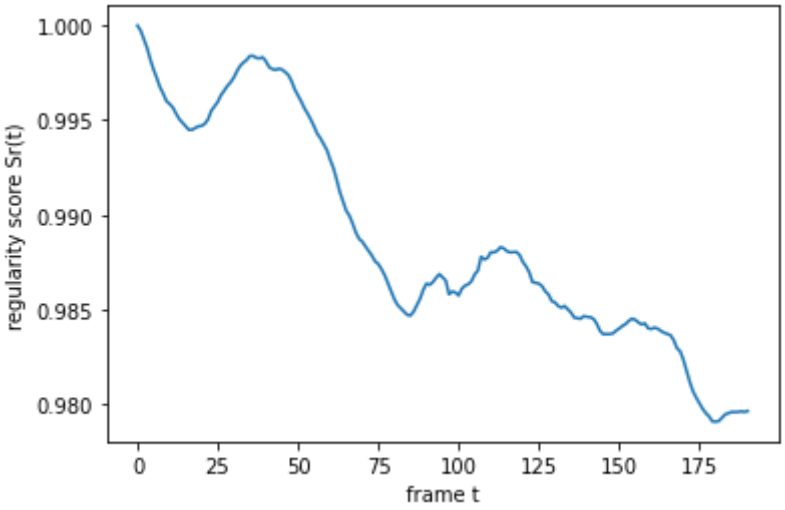



```
AUC:  0.599342105263158
EER:  0.4791666666666667
EER THRESHOLD:  0.9718371993276782
Optimal threshold value is: 0.9891555813008066
```

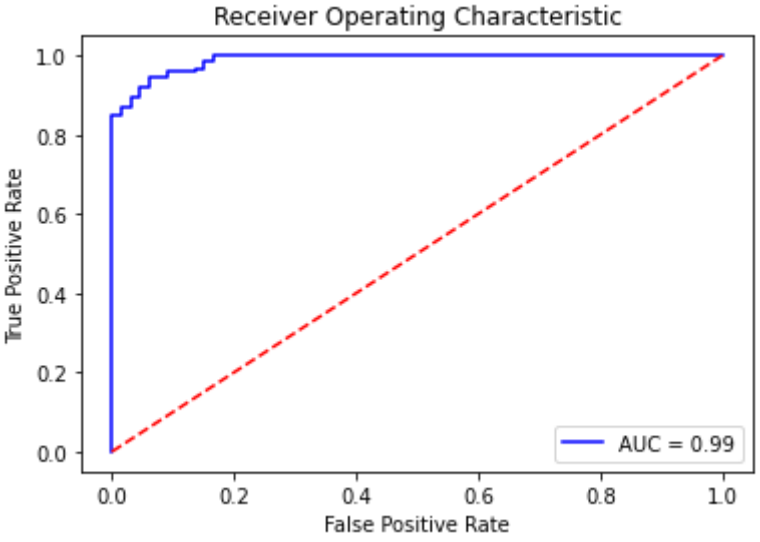


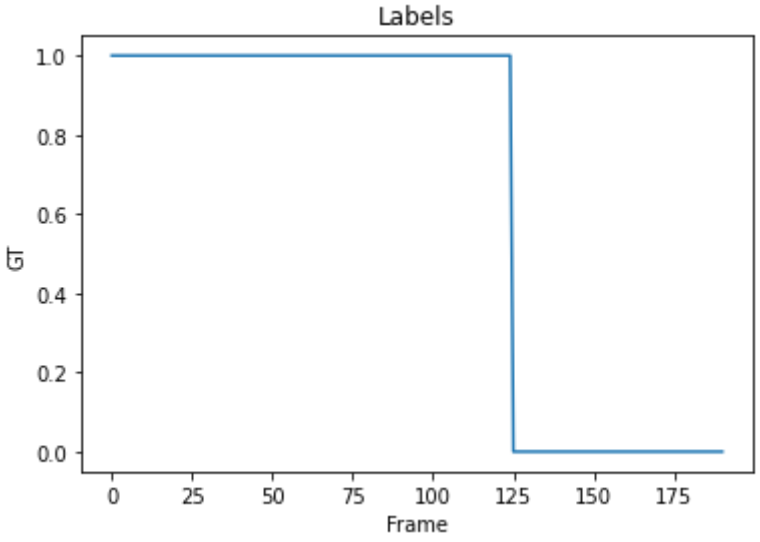



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test012
GT:  12
got model
(200, 227, 227, 1)
got data
```

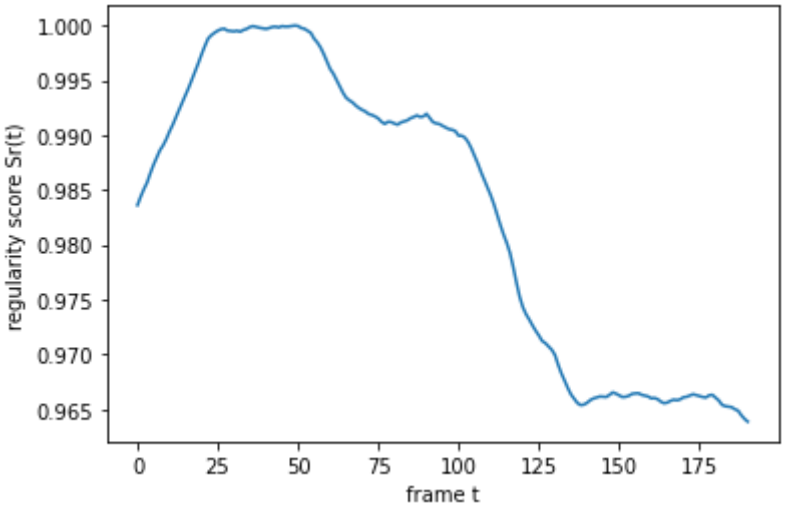



```
AUC:  0.9887272727272728
EER:  0.06060606060606061
EER THRESHOLD:  0.985536647717877
Optimal threshold value is: 0.985536647717877
```

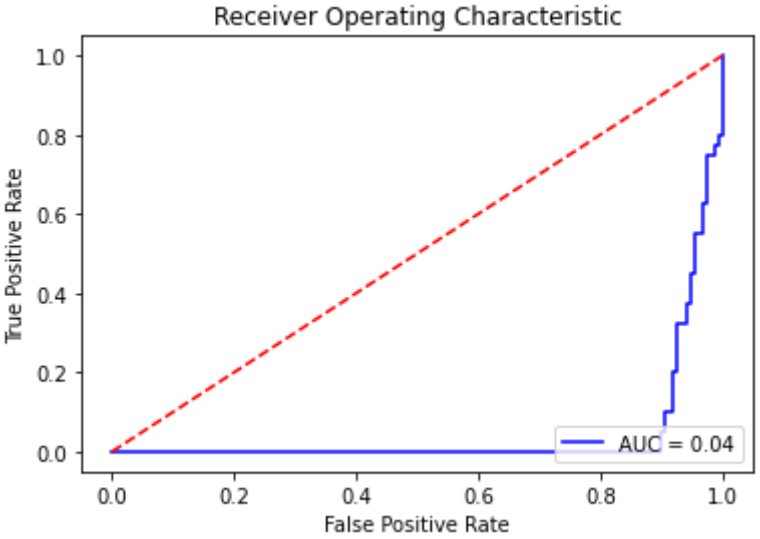


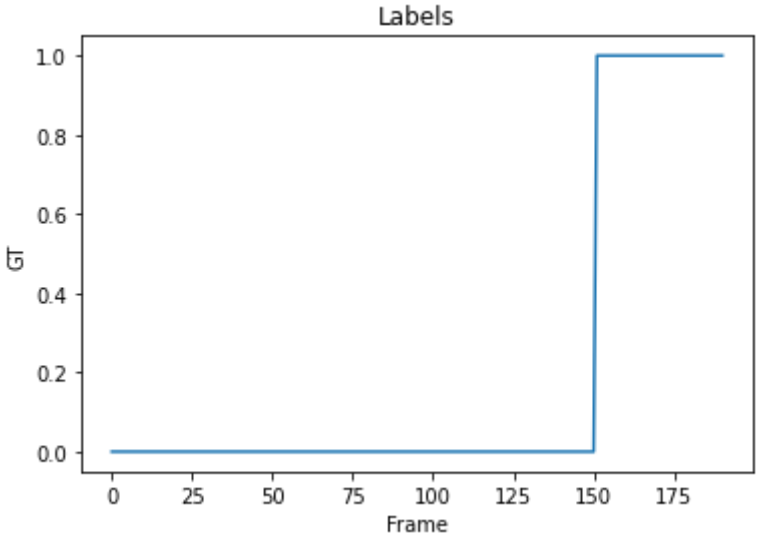



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test013
GT:  13
got model
(200, 227, 227, 1)
got data
```

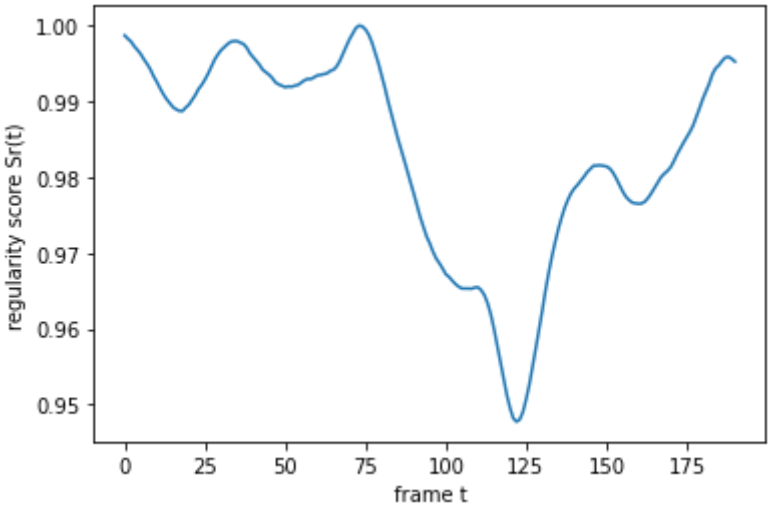



```
AUC:  0.04453642384105961
EER:  0.9072847682119205
EER THRESHOLD:  0.9663545848639332
Optimal threshold value is: 2.0
```

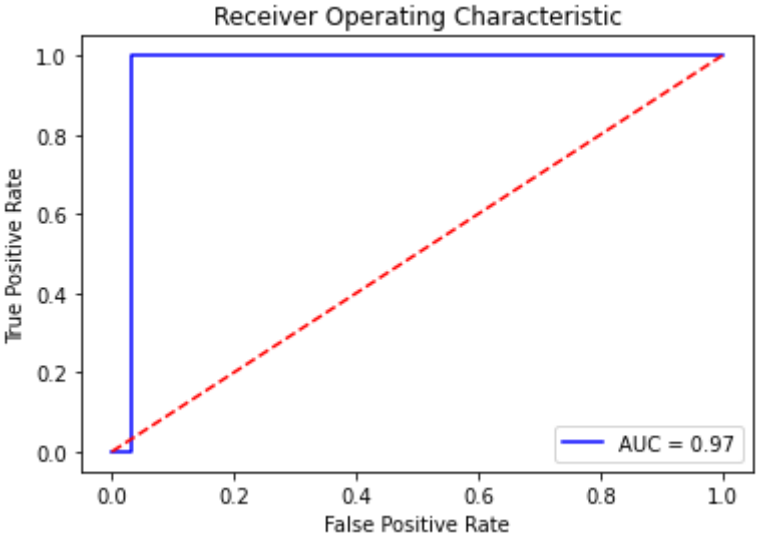


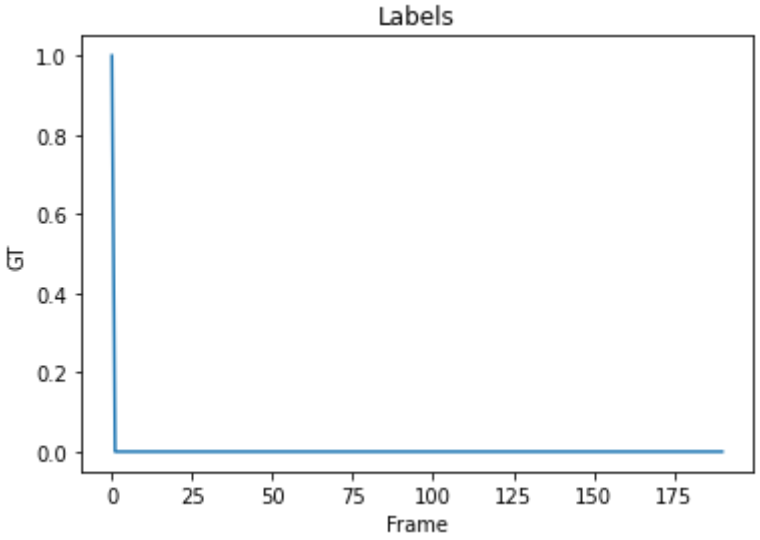



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test014
GT:  14
got model
(200, 227, 227, 1)
got data
```

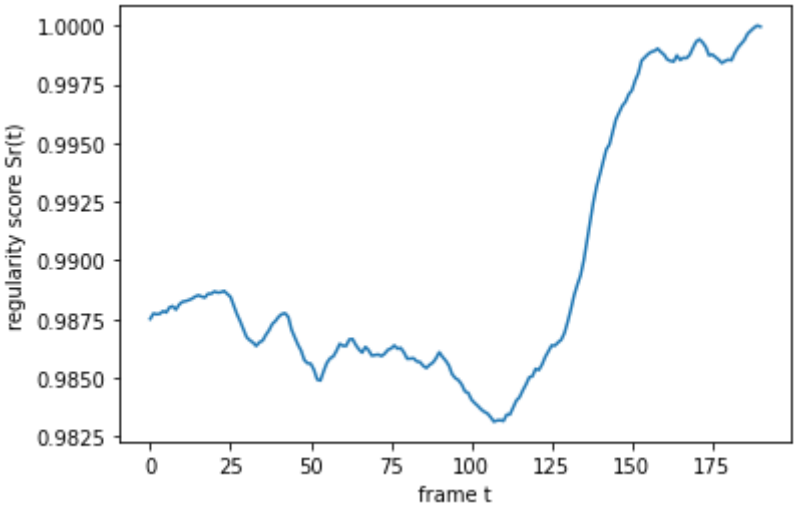



```
AUC:  0.968421052631579
EER:  0.031578947368421054
EER THRESHOLD:  0.9986948407674288
Optimal threshold value is: 0.9986948407674288
```

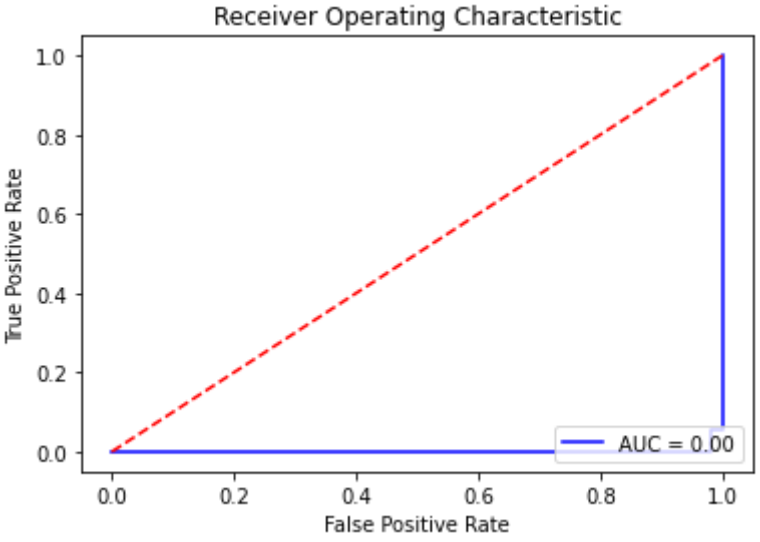


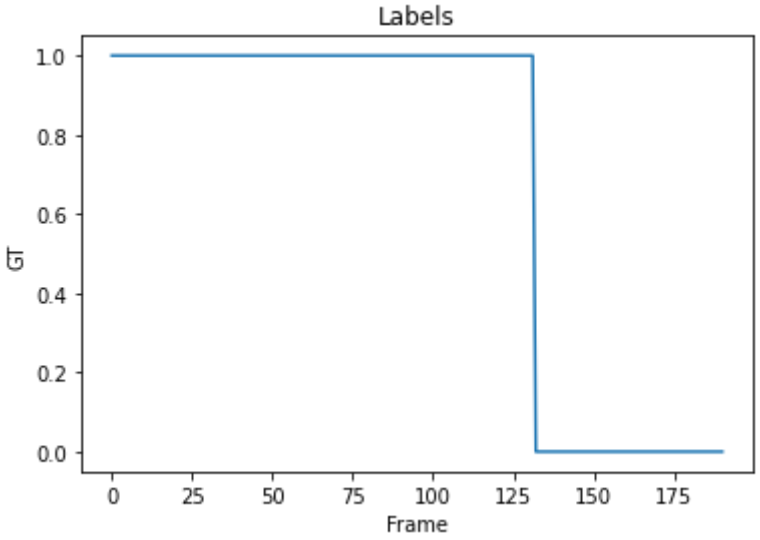



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test015
GT:  15
got model
(200, 227, 227, 1)
got data
```

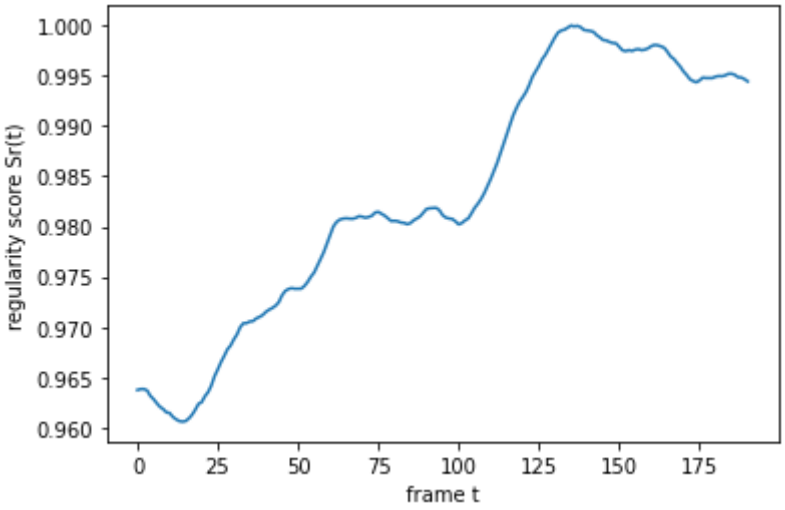



```
AUC:  0.0008988186954288625
EER:  0.9830508474576272
EER THRESHOLD:  0.9889676484484694
Optimal threshold value is: 2.0
```

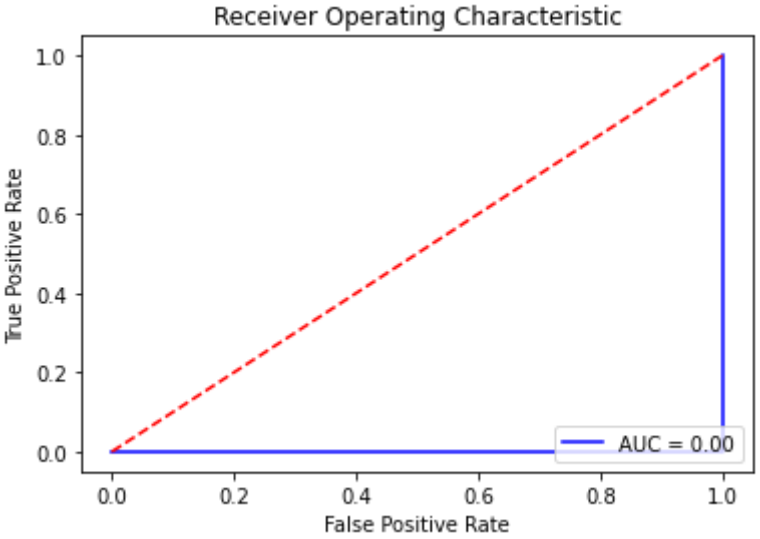


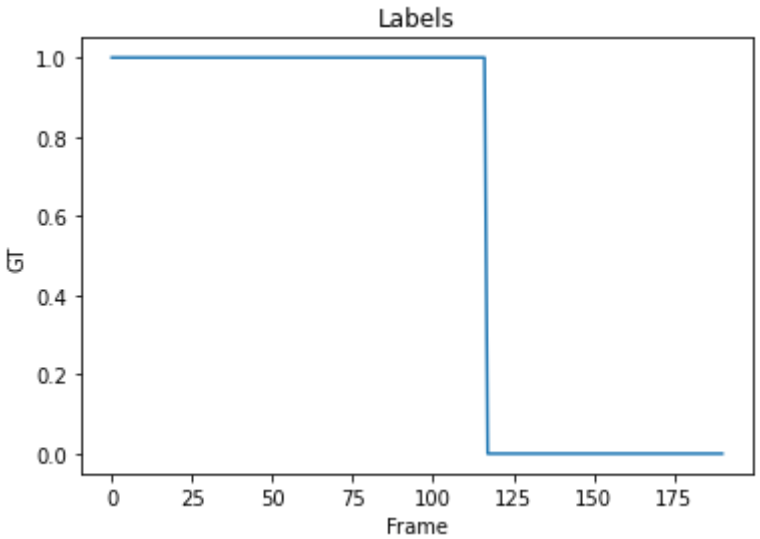



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test016
GT:  16
got model
(200, 227, 227, 1)
got data
```

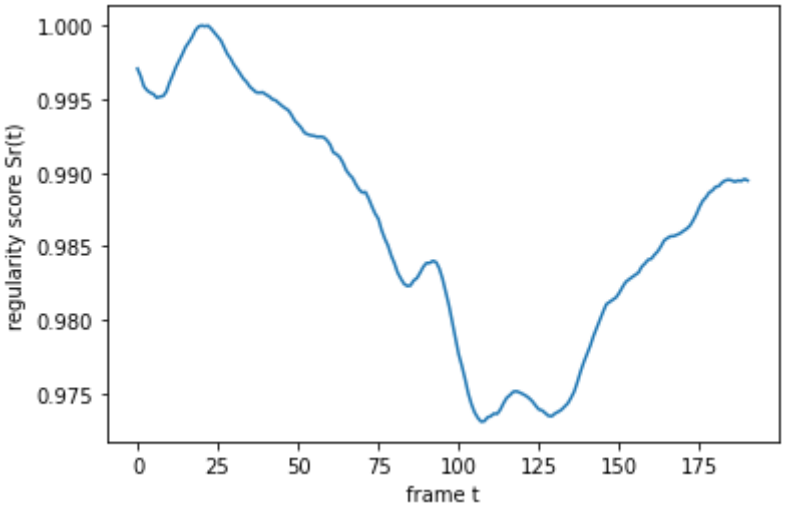



```
AUC:  0.0
EER:  1.0
EER THRESHOLD:  0.9910772563686646
Optimal threshold value is: 2.0
```

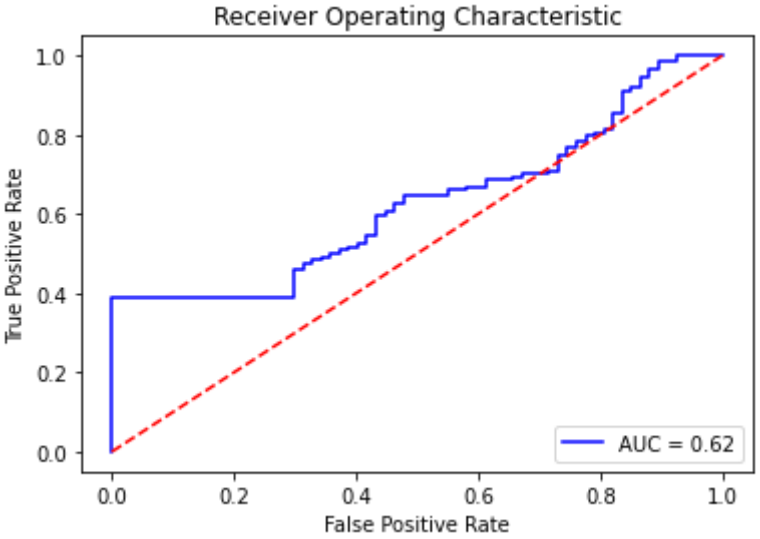


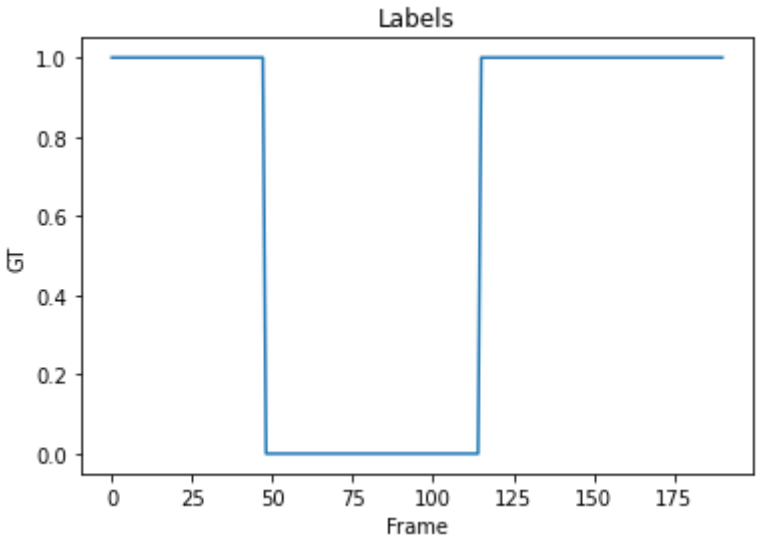



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test018
GT:  18
got model
(200, 227, 227, 1)
got data
```

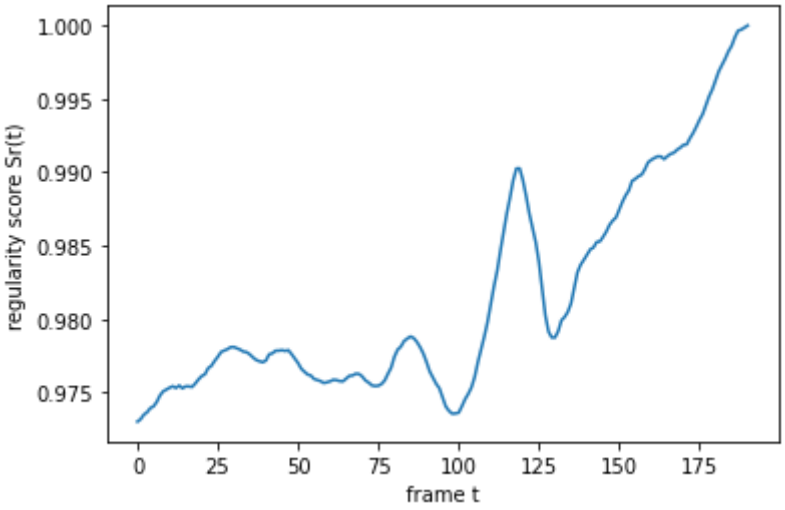



```
AUC:  0.624097255657198
EER:  0.43283582089552236
EER THRESHOLD:  0.9861222383295197
Optimal threshold value is: 0.9942466929132268
```

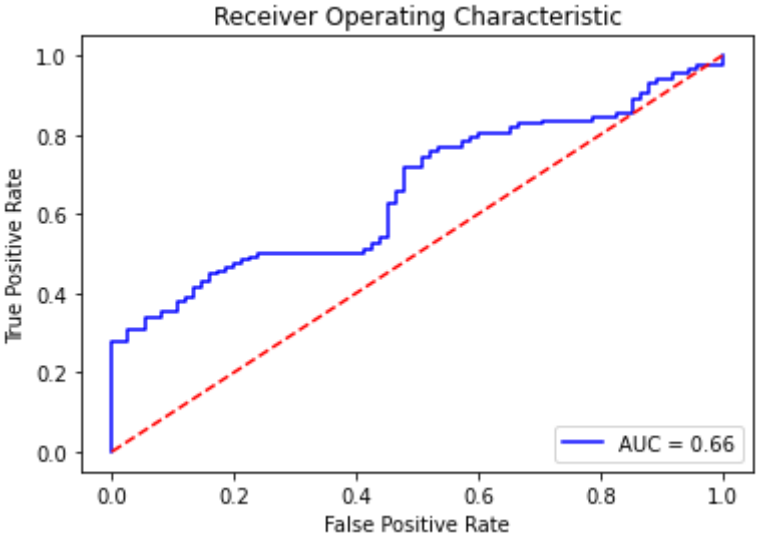


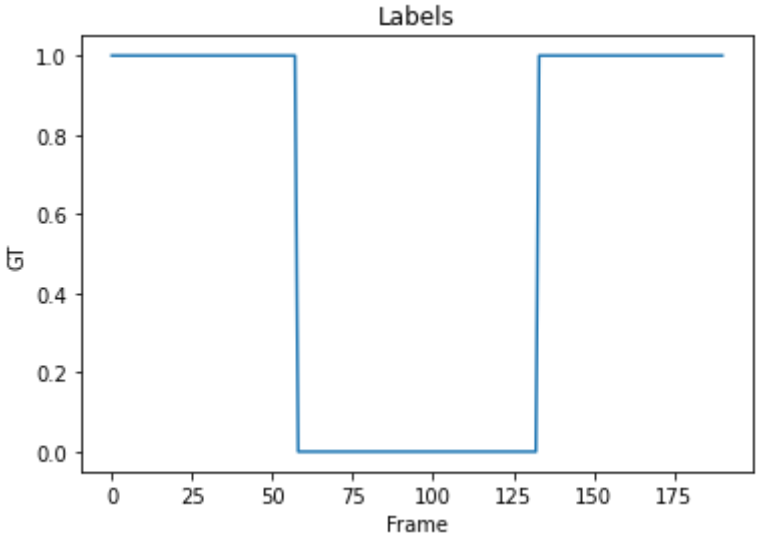



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test019
GT:  19
got model
(200, 227, 227, 1)
got data
```

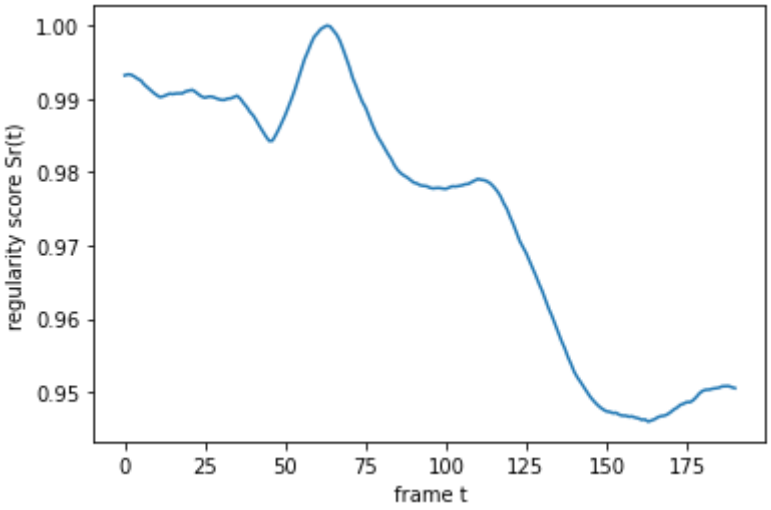



```
AUC:  0.6577011494252873
EER:  0.4533333333333333
EER THRESHOLD:  0.9779069284176417
Optimal threshold value is: 0.9840364068357789
```

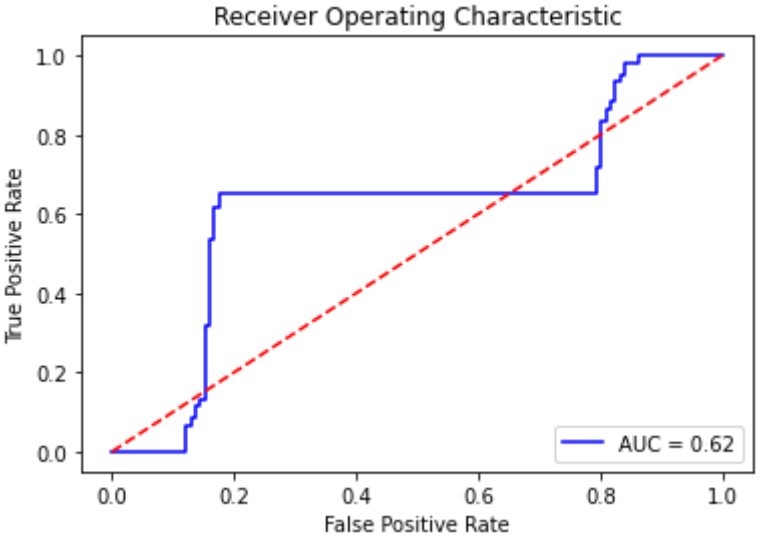


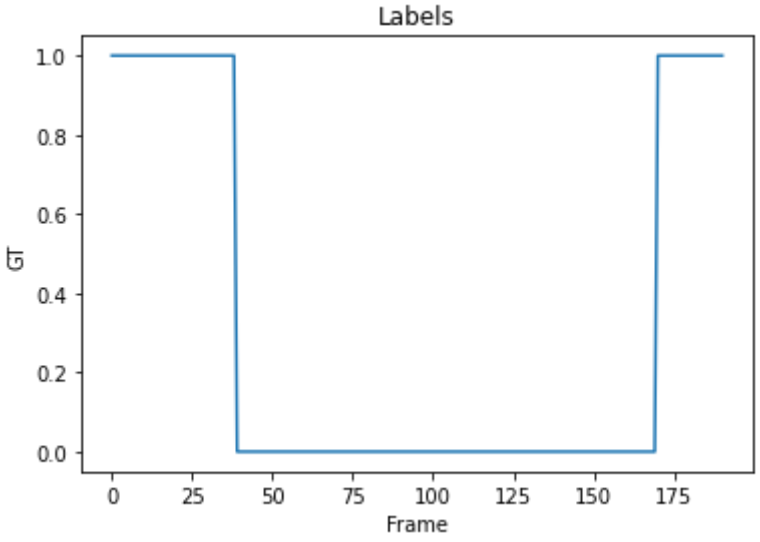



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test020
GT:  20
got model
(200, 227, 227, 1)
got data
```

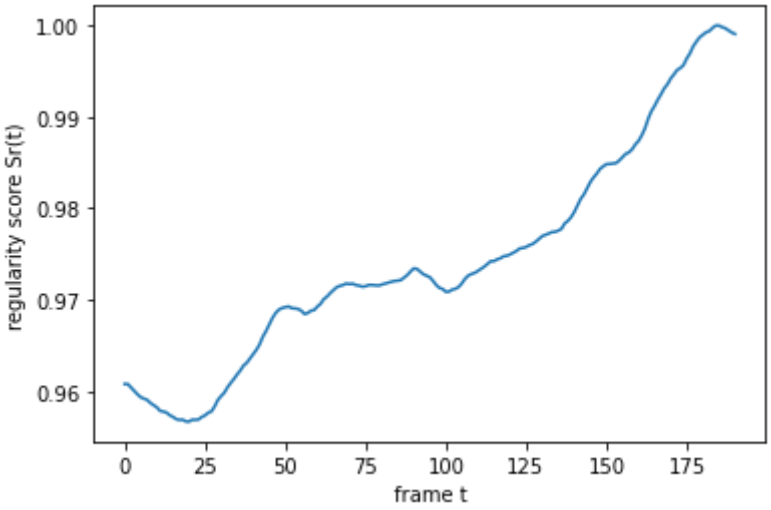



```
AUC:  0.6156488549618321
EER:  0.17557251908396945
EER THRESHOLD:  0.9889128250565253
Optimal threshold value is: 0.9889128250565253
```

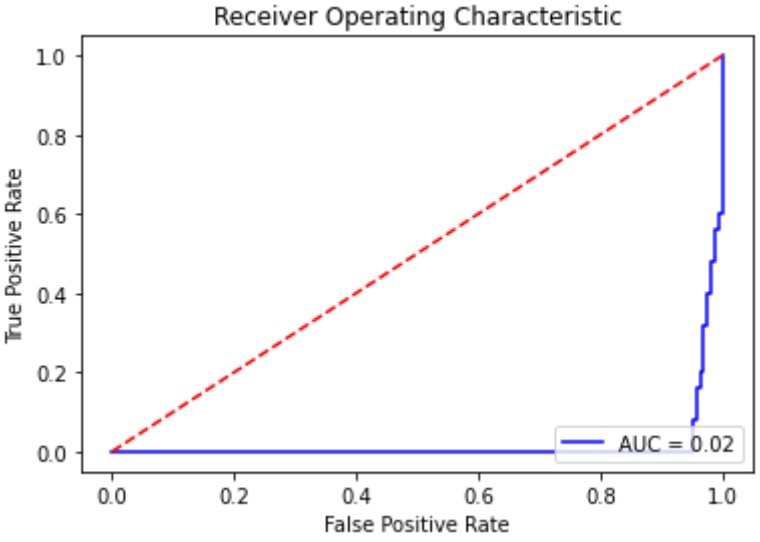


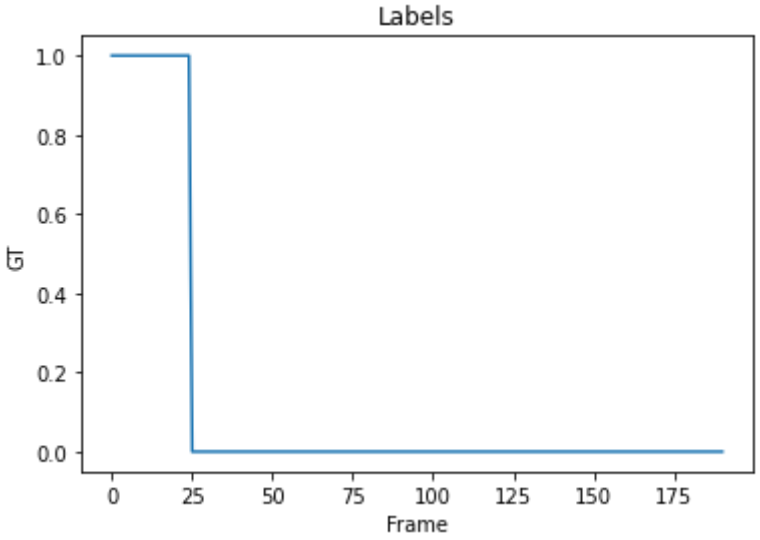



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test021
GT:  21
got model
(200, 227, 227, 1)
got data
```

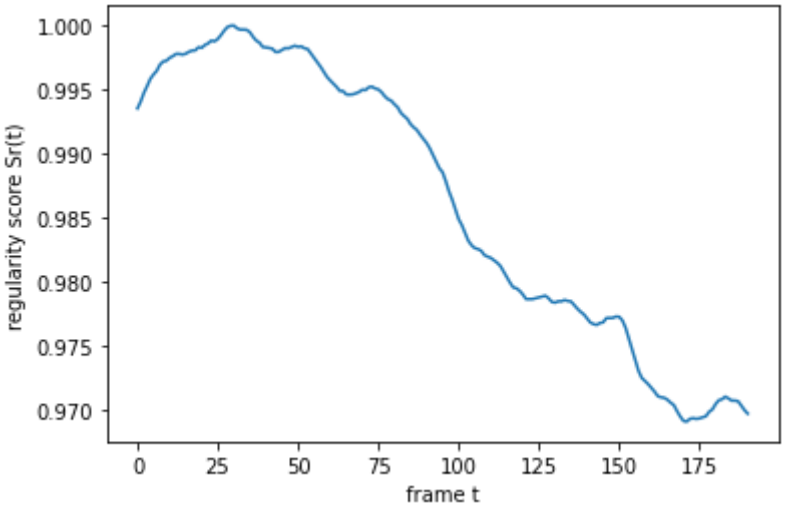



```
AUC:  0.01686746987951808
EER:  0.9518072289156626
EER THRESHOLD:  0.9607727955358614
Optimal threshold value is: 2.0
```

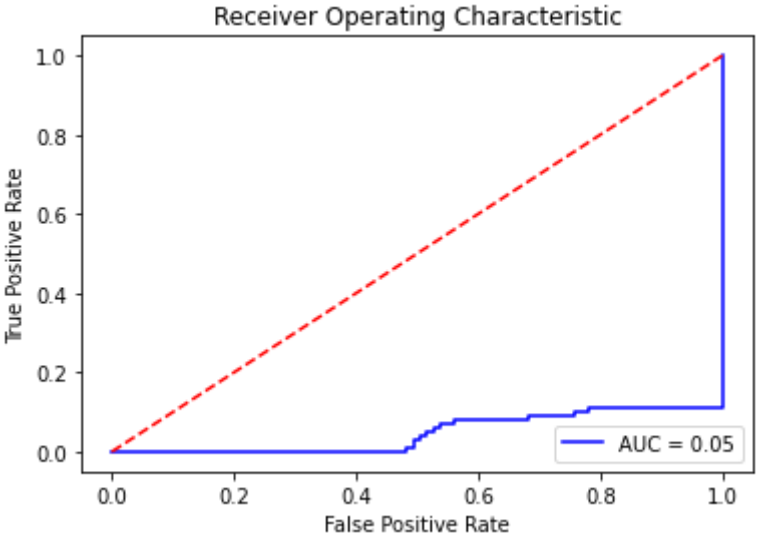


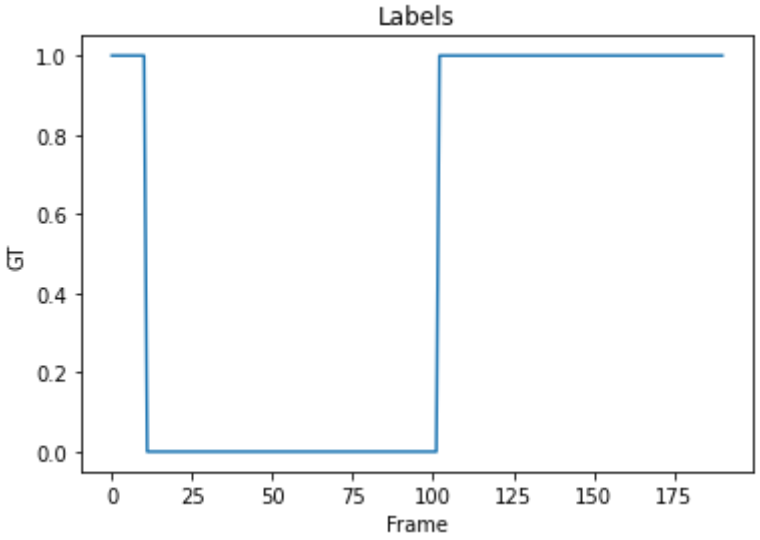



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test022
GT:  22
got model
(200, 227, 227, 1)
got data
```

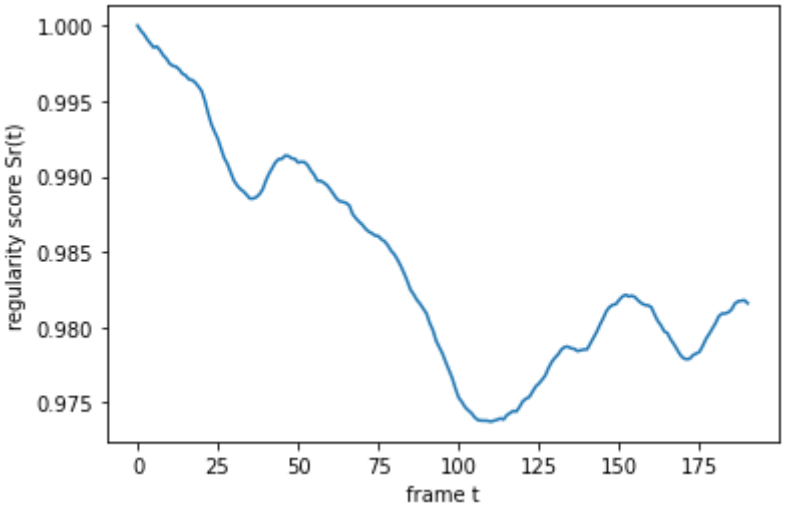



```
AUC:  0.04659340659340659
EER:  0.7802197802197802
EER THRESHOLD:  0.9935381649946942
Optimal threshold value is: 2.0
```

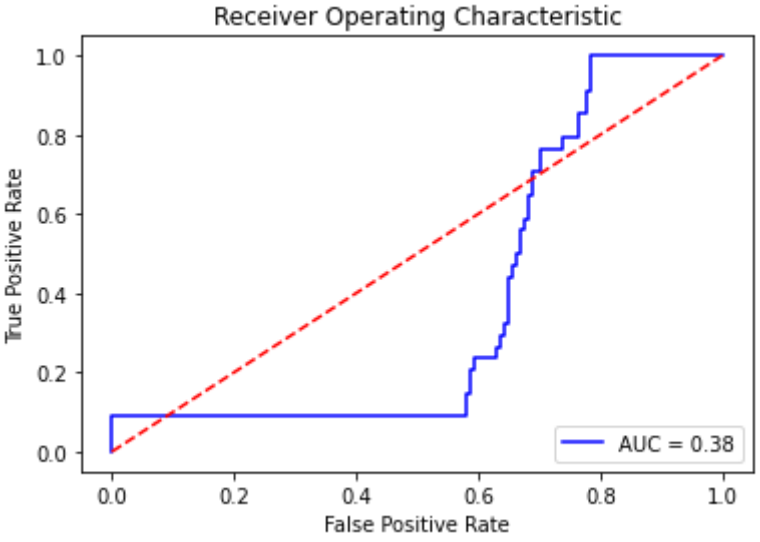


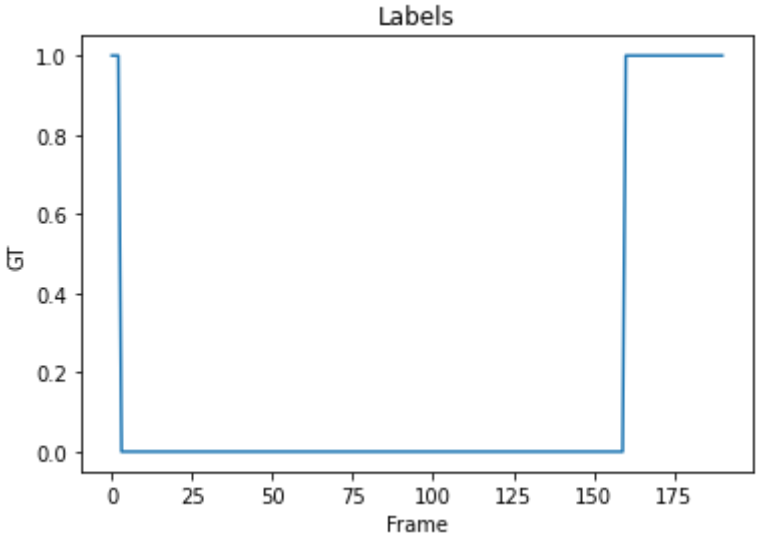



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test023
GT:  23
got model
(200, 227, 227, 1)
got data
```

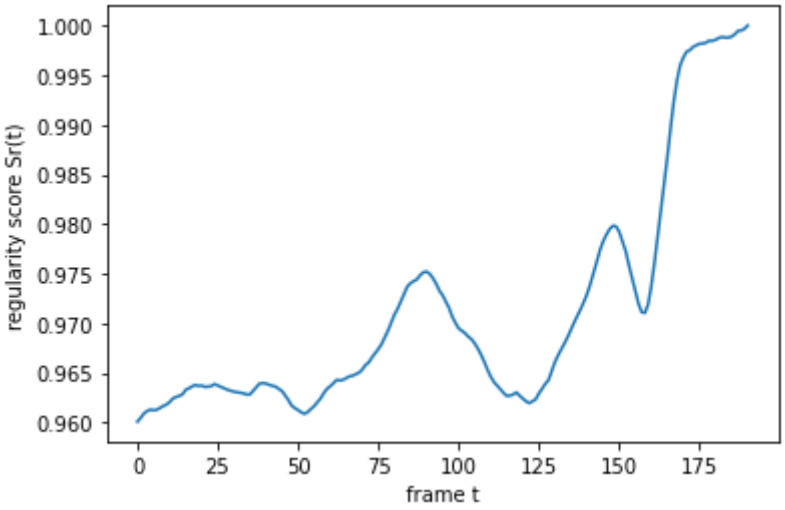



```
AUC:  0.38010490820532034
EER:  0.6496815286624203
EER THRESHOLD:  0.9809119644657353
Optimal threshold value is: 0.9778768269049793
```

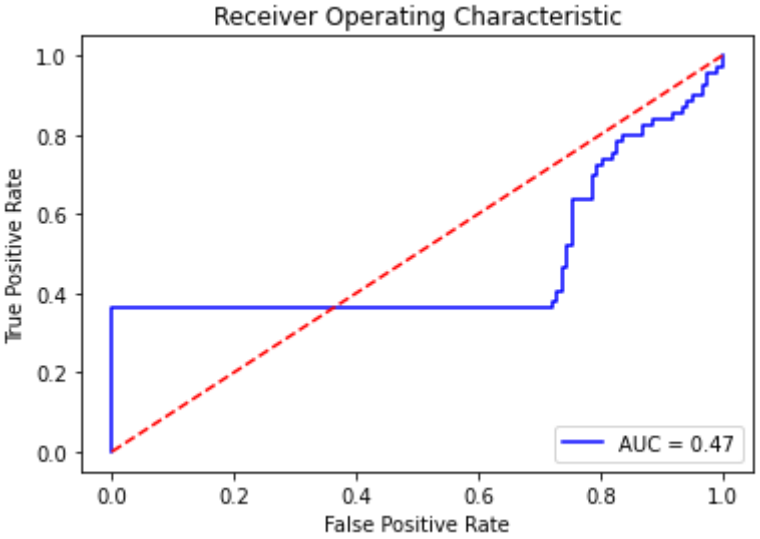


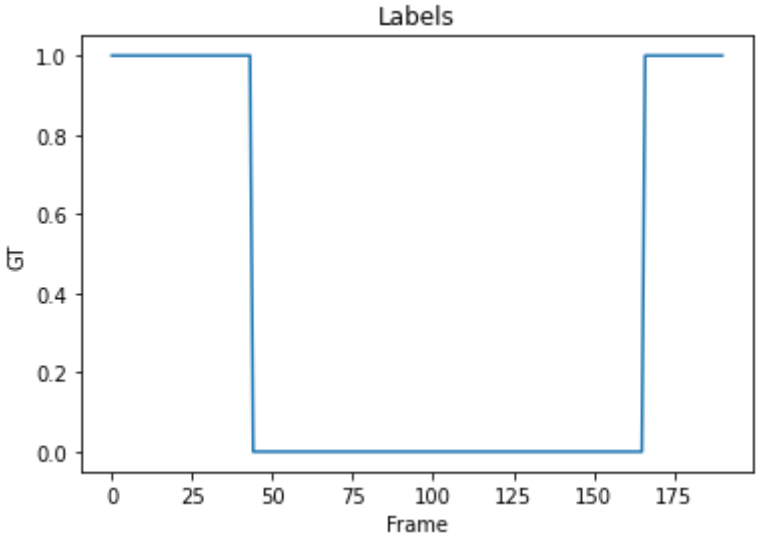



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test024
GT:  24
got model
(200, 227, 227, 1)
got data
```

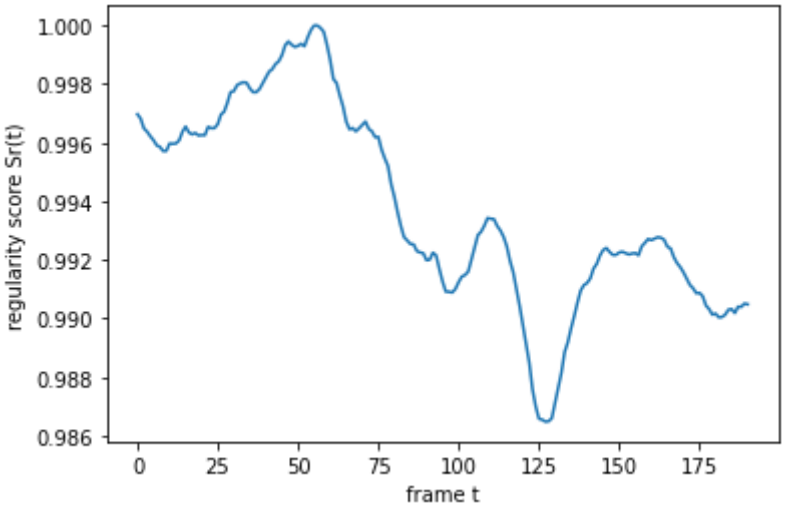



```
AUC:  0.4746970776906629
EER:  0.7213114754098361
EER THRESHOLD:  0.9640950464500151
Optimal threshold value is: 0.9896871564040846
```

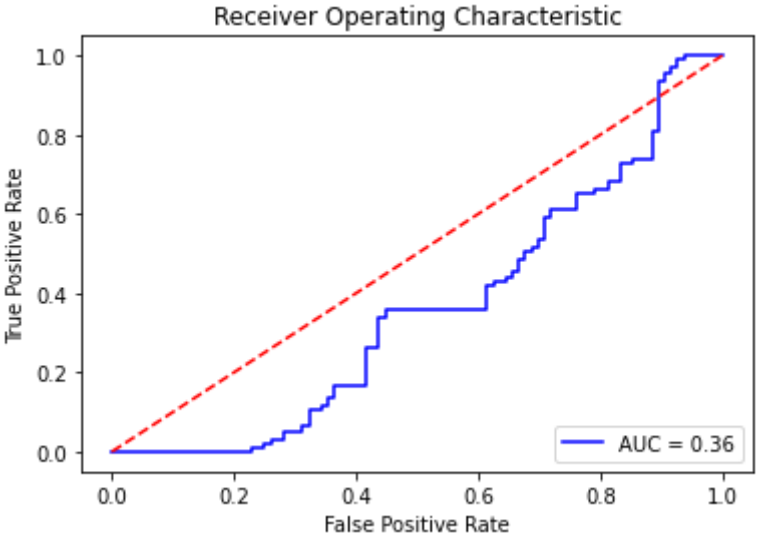


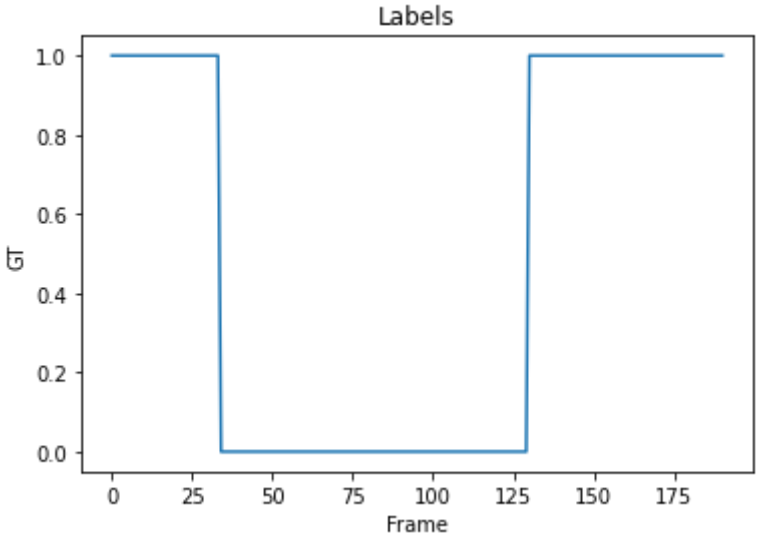



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test025
GT:  25
got model
(200, 227, 227, 1)
got data
```

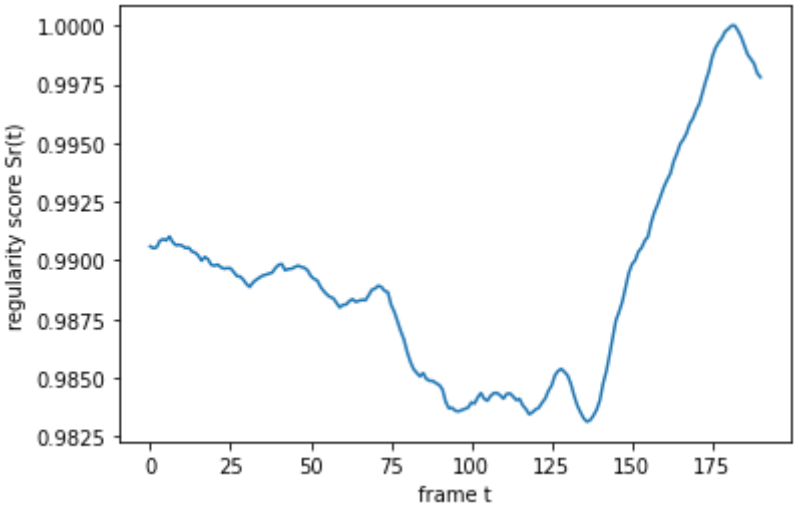



```
AUC:  0.35888157894736844
EER:  0.6145833333333334
EER THRESHOLD:  0.9927814902962568
Optimal threshold value is: 0.9871081905158792
```

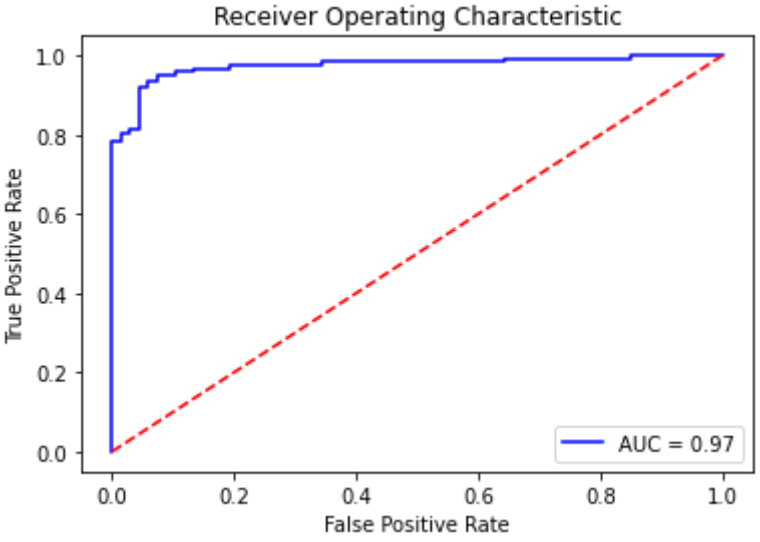


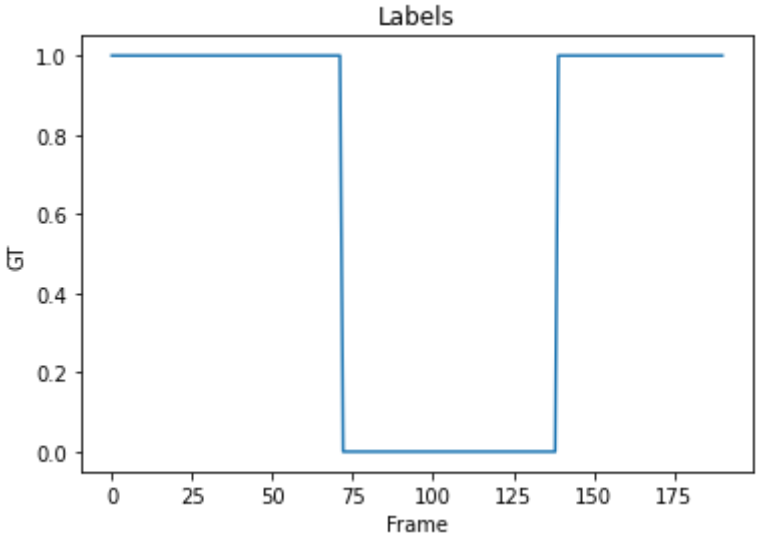



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test026
GT:  26
got model
(200, 227, 227, 1)
got data
```

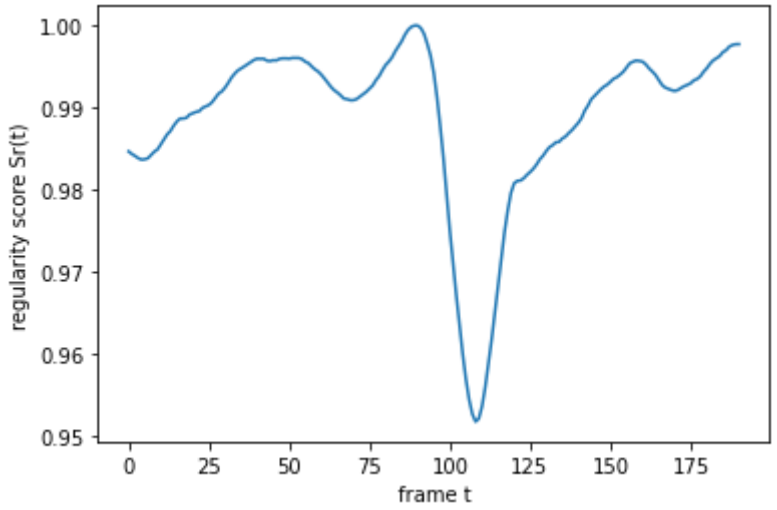



```
AUC:  0.9742416947520463
EER:  0.05970149253731343
EER THRESHOLD:  0.9879879936005234
Optimal threshold value is: 0.9874390613670057
```

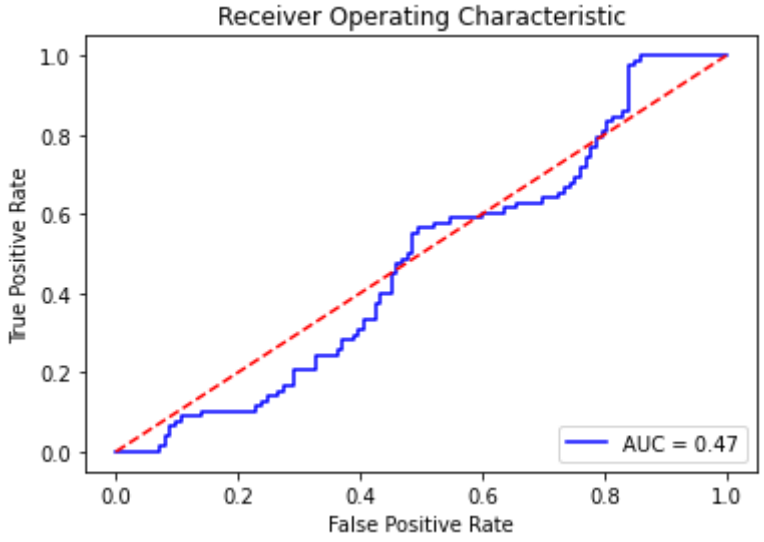


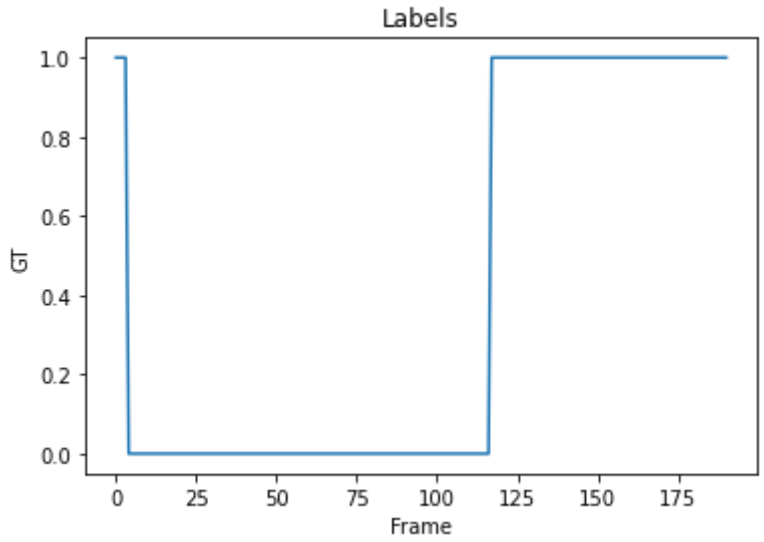



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test027
GT:  27
got model
(200, 227, 227, 1)
got data
```

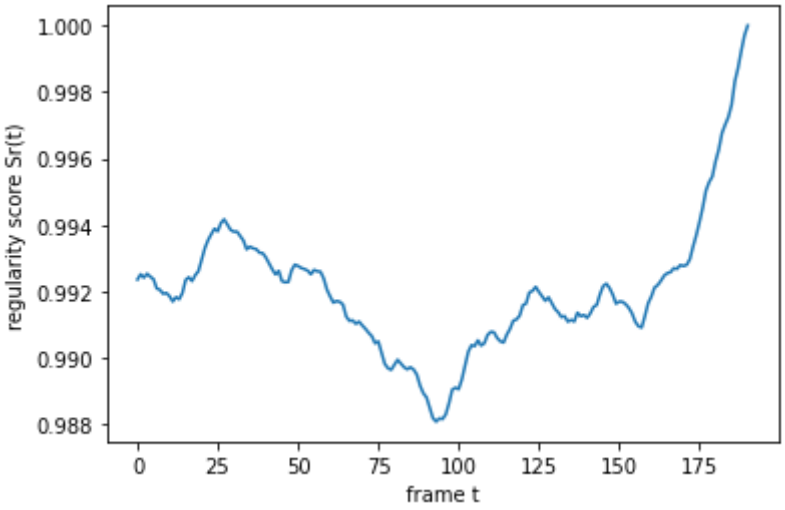



```
AUC:  0.4746993419559791
EER:  0.48672566371681414
EER THRESHOLD:  0.9923501795284171
Optimal threshold value is: 0.974833786603235
```

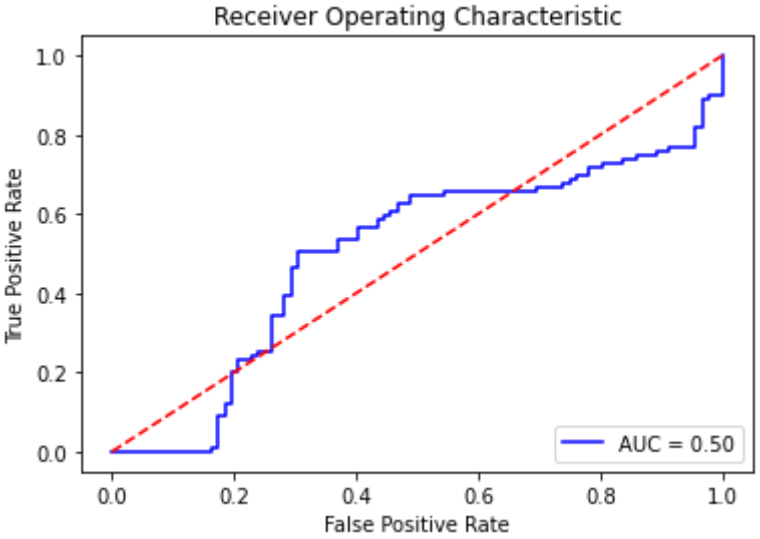


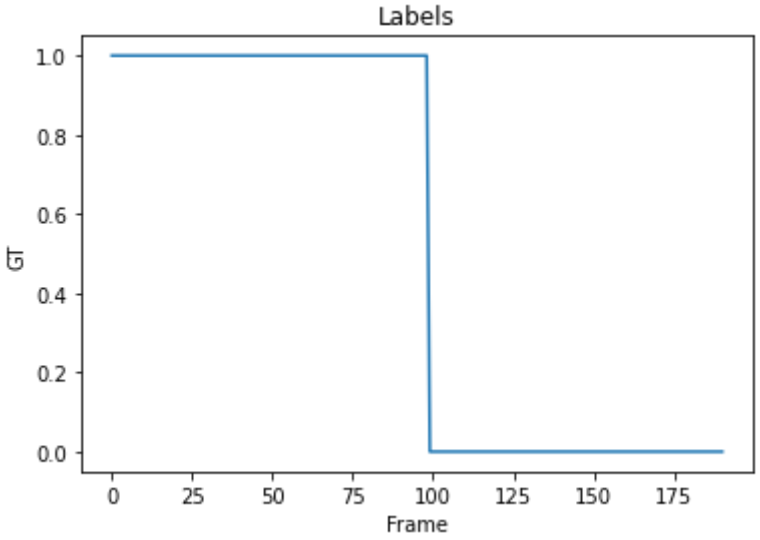



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test028
GT:  28
got model
(200, 227, 227, 1)
got data
```

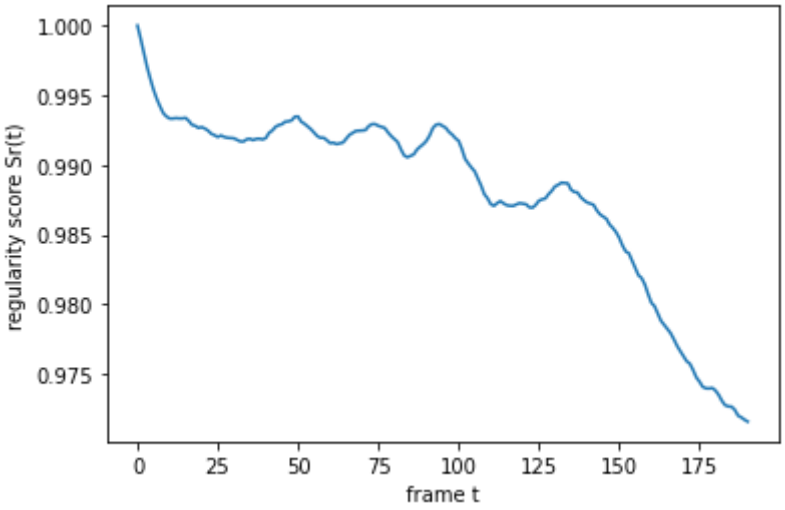



```
AUC:  0.49813350900307424
EER:  0.43478260869565216
EER THRESHOLD:  0.9918572276842792
Optimal threshold value is: 0.9922758141244057
```

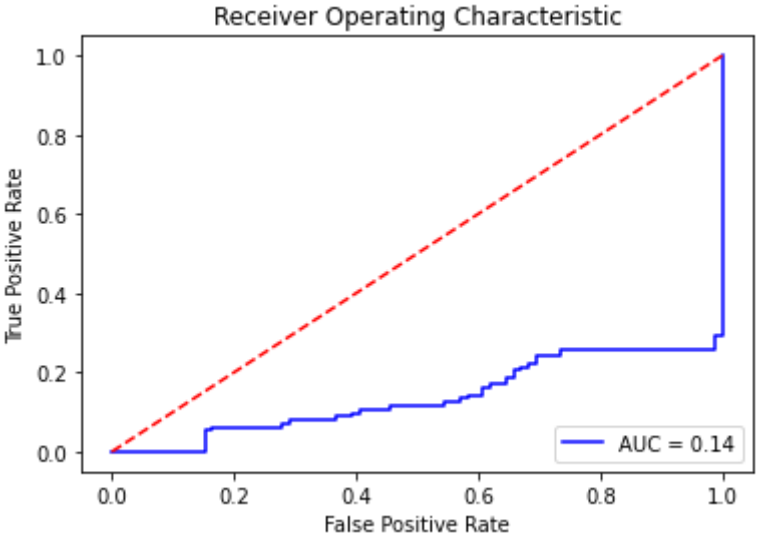


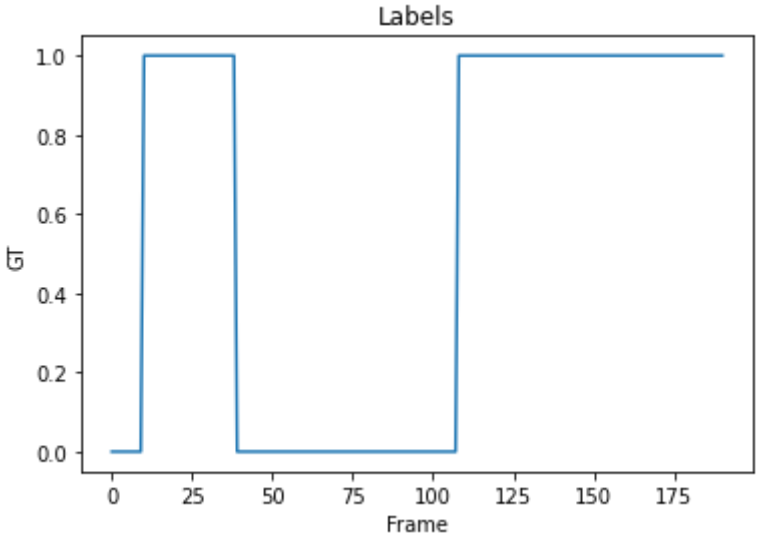



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test029
GT:  29
got model
(200, 227, 227, 1)
got data
```

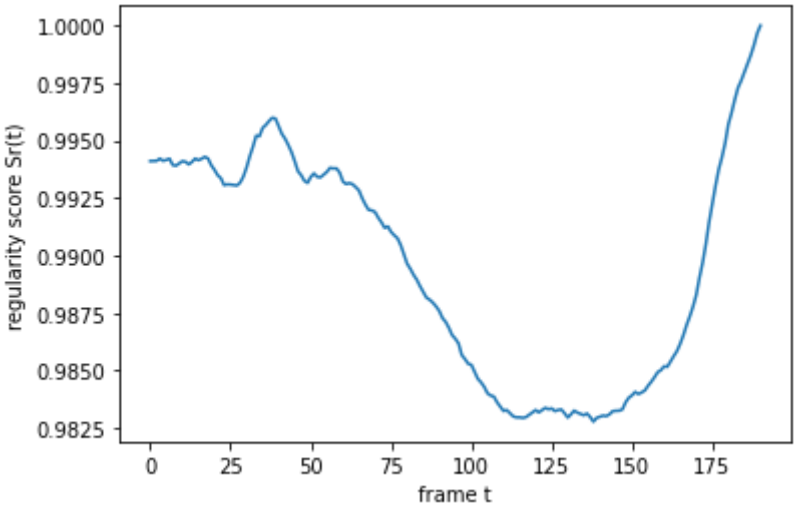



```
AUC:  0.1377712477396022
EER:  0.7341772151898734
EER THRESHOLD:  0.9916745113488104
Optimal threshold value is: 2.0
```

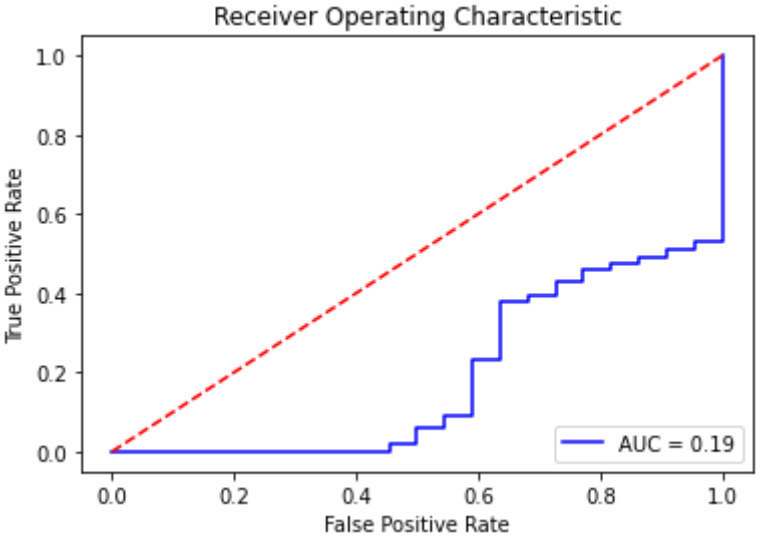


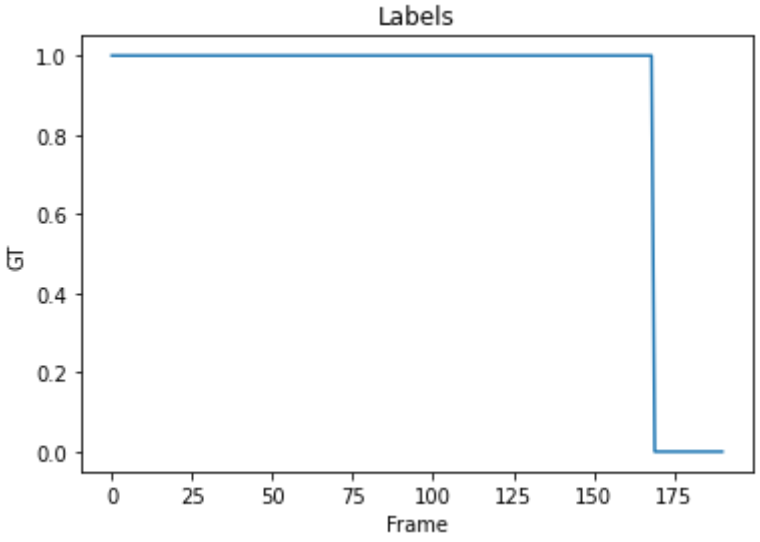



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test030
GT:  30
got model
(200, 227, 227, 1)
got data
```

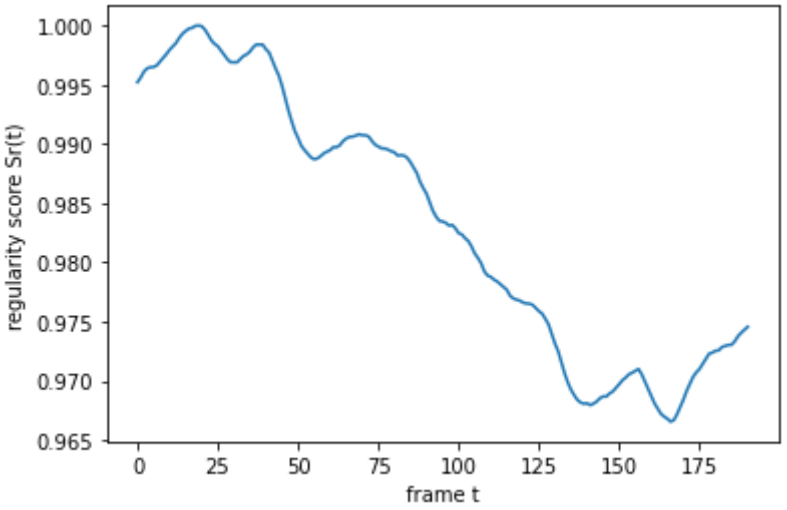



```
AUC:  0.18504572350726198
EER:  0.6363636363636364
EER THRESHOLD:  0.993046724482617
Optimal threshold value is: 2.0
```

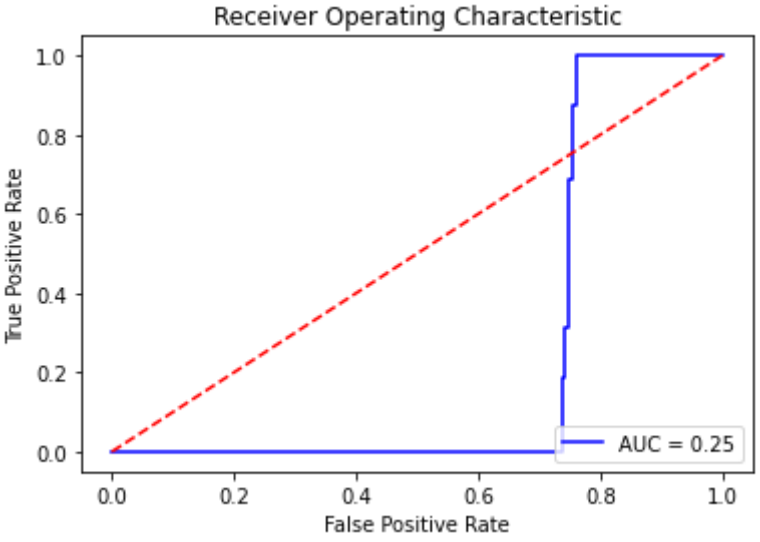


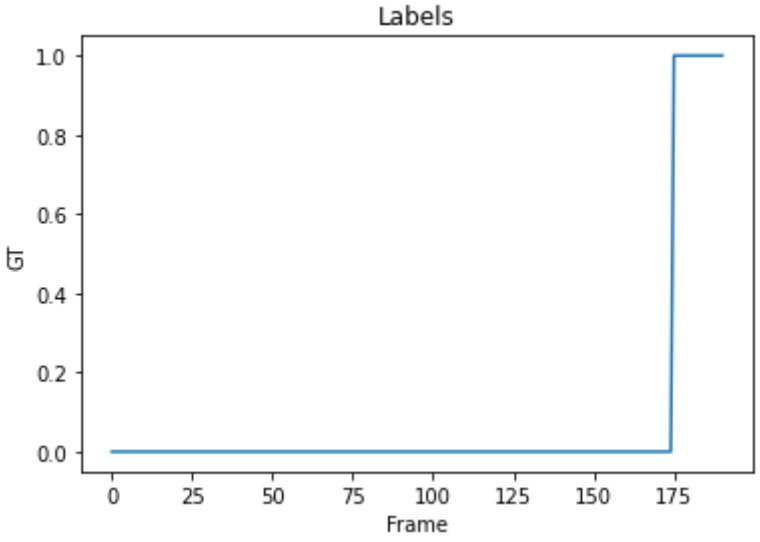



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test031
GT:  31
got model
(200, 227, 227, 1)
got data
```

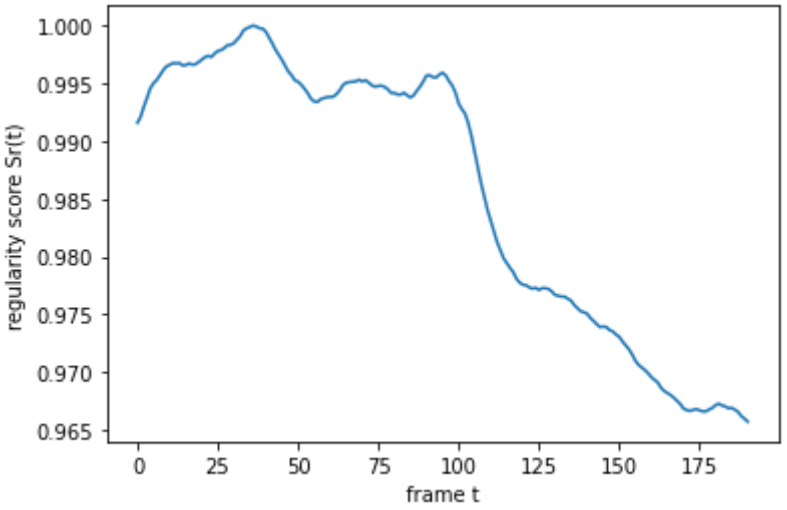



```
AUC:  0.2517857142857143
EER:  0.7428571428571429
EER THRESHOLD:  0.9733540580411165
Optimal threshold value is: 0.9710100813488566
```

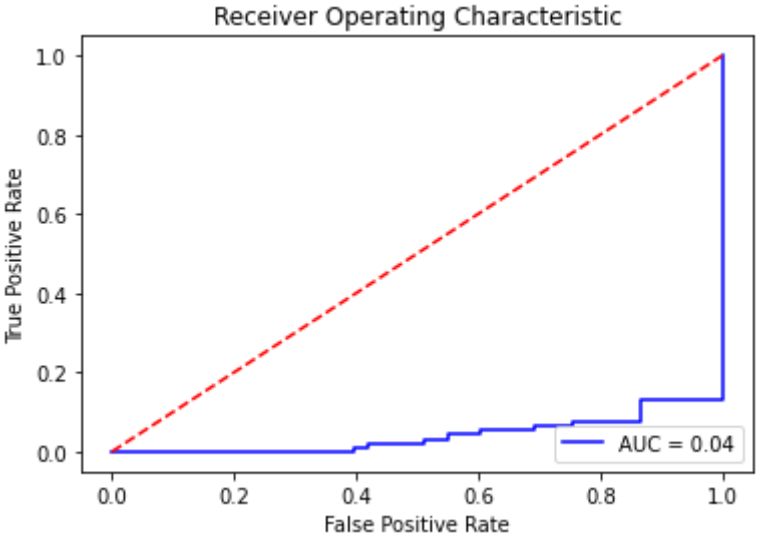


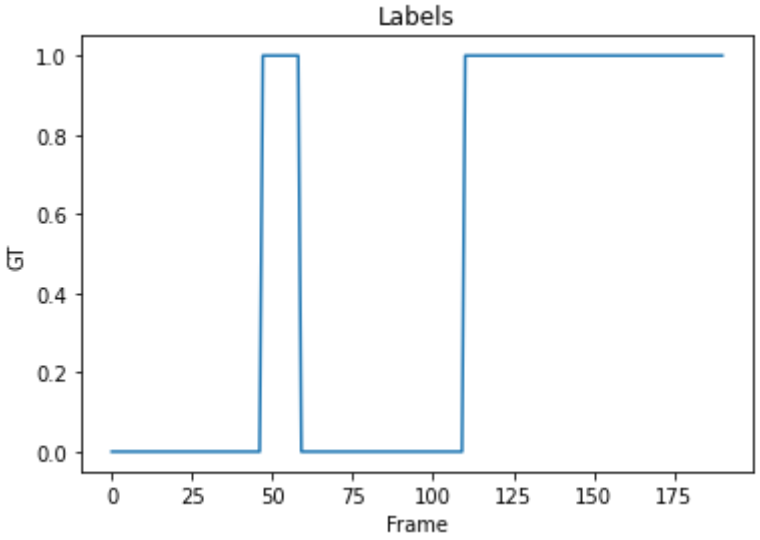



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test032
GT:  32
got model
(200, 227, 227, 1)
got data
```

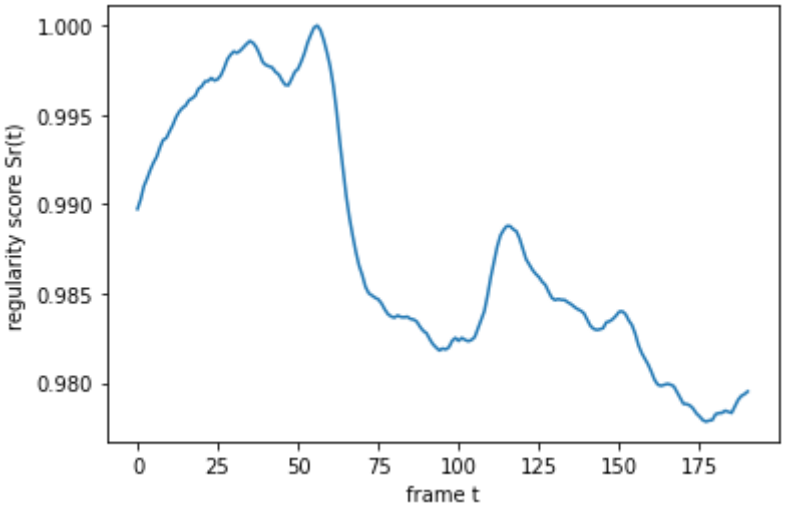



```
AUC:  0.04015799868334431
EER:  0.8673469387755102
EER THRESHOLD:  0.993384865227258
Optimal threshold value is: 2.0
```

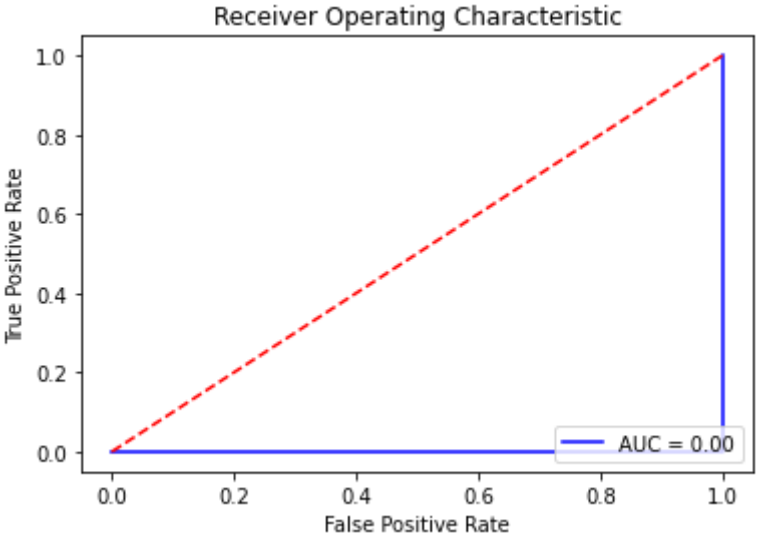


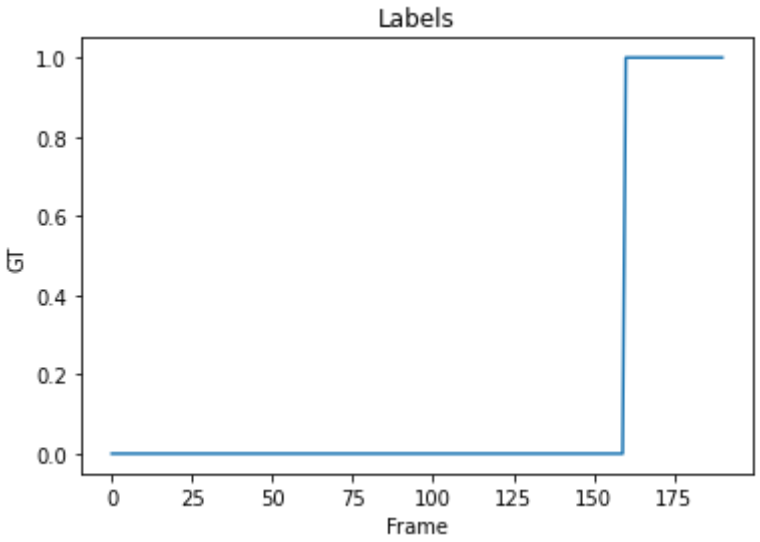



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test033
GT:  33
got model
(200, 227, 227, 1)
got data
```

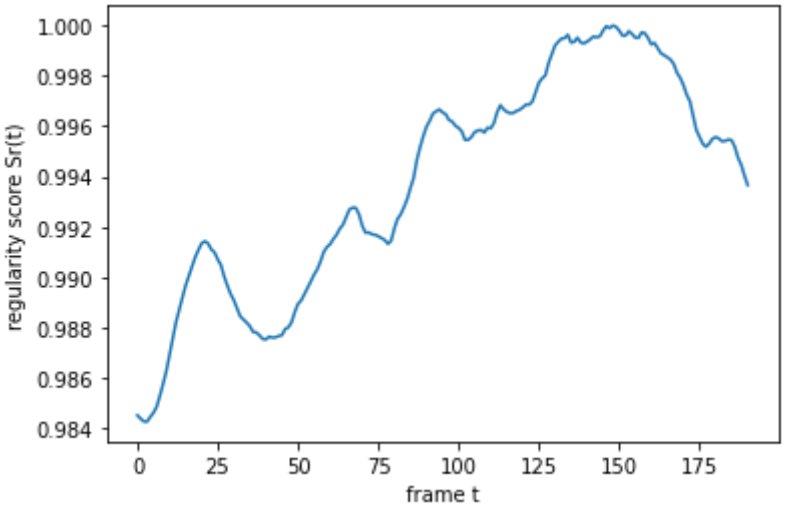



```
AUC:  0.0
EER:  1.0
EER THRESHOLD:  0.9810275185539563
Optimal threshold value is: 2.0
```

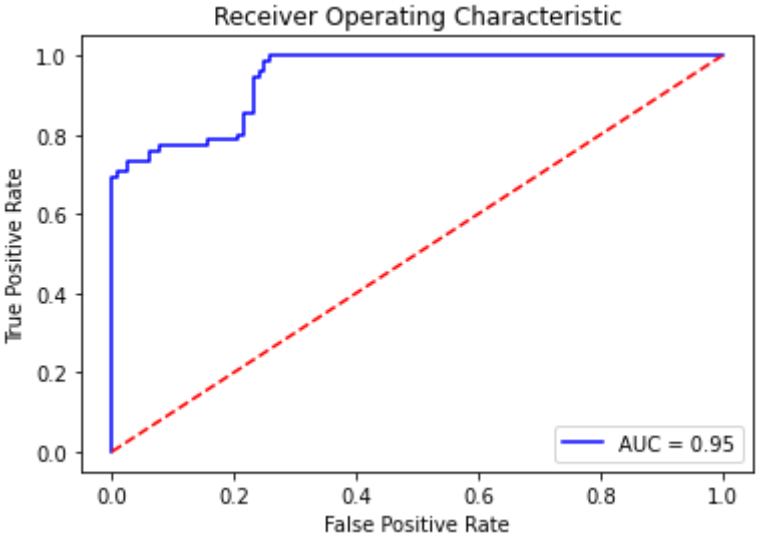


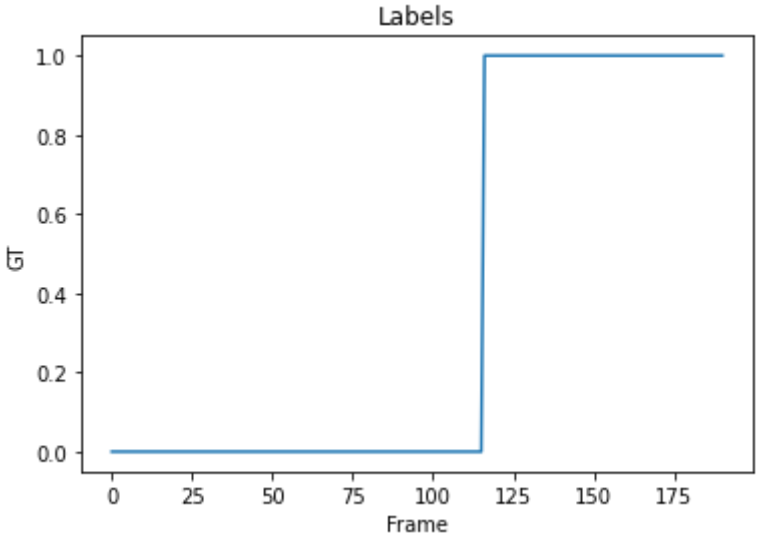



```
PATH:  UCSD_v5/UCSD_Anomaly_Dataset.v1p2/UCSDped1/Test/Test034
GT:  34
got model
(200, 227, 227, 1)
got data
```

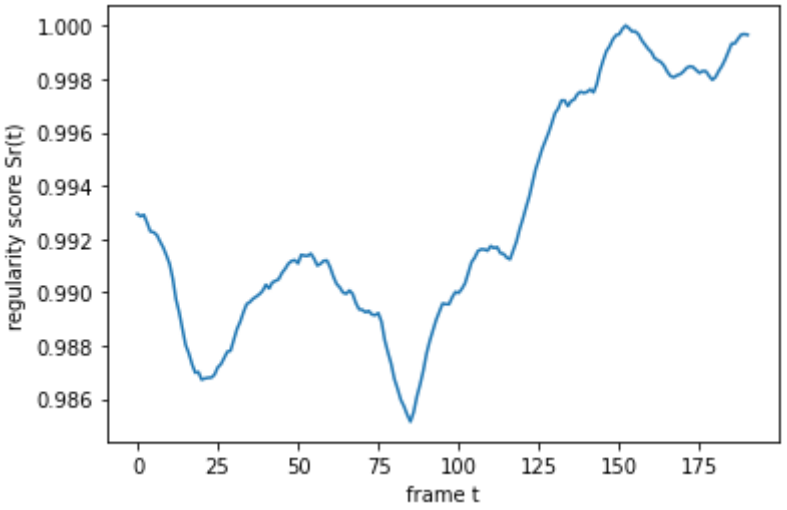



```
AUC:  0.9451724137931035
EER:  0.20689655172413793
EER THRESHOLD:  0.9956070829292043
Optimal threshold value is: 0.9936604660760492
```






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




```
AUC:  0.18175675675675673
EER:  0.7747747747747747
EER THRESHOLD:  0.991175717557958
Optimal threshold value is: 0.9867285939723841
```

Receiver Operating Characteristic

In [ ]: