```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473189

Enter your authorization code:
··········
Mounted at /content/drive
```

Importing libraries

```
import os #provides functions to interact with file system
import cv2 #openCV library
import numpy as np #array processing package in python
import matplotlib.pyplot as plt #use to plot graphs
import shutil #offers high level operations on files and collection of files
import sys #provides information about constants, functions and methods
print(sys.version)
#from PIL import Image
#from keras.preprocessing.image import img_to_array,load_img
```

```
3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0]
```

Changing directory to Major Project directory

```
print(os.getcwd())
os.chdir('drive/My Drive/Major Project')
print(os.getcwd())
```

```
/content
/content/drive/My Drive/Major Project
```

```
print(os.listdir())
```

```
['Avenue Dataset', 'UCSDped1', 'TestingData1', 'TestingData2', 'TestingData3', 'TestingD
```

Convert each training video to sequence of frames

```
def createTrainingDataPerVideo( i ):
  currentframe = 0
  count = 0
  try:
    # creating a folder named TrainingData1
    if not os.path.exists( 'TrainingData'+ str(i) ):
      os.makedirs( 'TrainingData' + str(i) )
    # if not created then raise error
  except OSError:
      print ('Error: Creating directory of data')
  videoPath = "Avenue Dataset/training_videos/{}.avi".format(i)
  # Create VideoCapture object to capture a video from specified path.
  cam = cv2.VideoCapture(videoPath)
  # frames extraction from video
  while(True):
    # Sets position of video file to read every frame after 120 ms or 0.12 s.
    cam.set(cv2.CAP_PROP_POS_MSEC,(currentframe*120))
    # Capture frame by frame...ret stores true if frame is read, otherwise false
    ret,frame = cam.read()
    if ret:
        # if video is still left continue creating images
        name = './TrainingData' + str(i) + '/frame' + str(count) + '.jpg'
        print ('Creating...' + name)

        # write extracted images to current directory
        cv2.imwrite(name, frame)

        # increasing counter so that it will
        # show how many frames are created
        currentframe += 1
        count += 1
    else:
        break
    # Release all space and windows once done
  cam.release()
  cv2.destroyAllWindows()
```

Create Training data

```
def createTrainingData():
  for k in range(1,17):
    createTrainingDataPerVideo(k)
createTrainingData()
print('Training Data Created')
```

Preprocessing of training data i.e., resize, rgb to gray, Normalization

```
def getTrainingData():
  imagestore = []
  for k in range(1,17):
    framepath = 'TrainingData'+str(k)
    images = os.listdir(framepath)
    print(len(images))
    print(len(images))
    for image in images:
      image_path=framepath+ '/'+ image
      img = cv2.imread(image_path)
      img = cv2.resize(img,(227,227))
      #plt.imshow(img)
      #Convert the Image to Grayscale
      gray = 0.2989*img[:,:,0] + 0.5870*img[:,:,1] + 0.1140*img[:,:,2]
      plt.imshow(gray, cmap = 'gray')
      imagestore.append(gray)
  print(len(imagestore))
  imagestore_array = np.array(imagestore)
  a,b,c=imagestore_array.shape
  print(a,b,c)
  #Reshape to (227,227,batch_size)
  imagestore_array.resize(b,c,a)
  print(imagestore_array.shape)
  #Normalize
  imagestore_array_normalize=(imagestore_array-imagestore_array.mean())/(imagestore_array.std
  #Clip negative Values
  imagestore_clip=np.clip(imagestore_array_normalize,0,1)
  return imagestore_clip
```

Create spatio-temporal autoencoder model

```
from keras.layers import Conv3D,ConvLSTM2D,Conv3DTranspose
from keras.models import Sequential
def lo_model():
  model=Sequential()
  model.add(Conv3D(filters=128,kernel_size=(11,11,1),strides=(4,4,1),padding='valid',input_sh
  model.add(Conv3D(filters=64,kernel_size=(5,5,1),strides=(2,2,1),padding='valid',activation=
  model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,padding='same',dropout=0.4,recu
  model.add(ConvLSTM2D(filters=32,kernel_size=(3,3),strides=1,padding='same',dropout=0.3,retu
  model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,return_sequences=True, padding=
  model.add(Conv3DTranspose(filters=128,kernel_size=(5,5,1),strides=(2,2,1),padding='valid',a
  model.add(Conv3DTranspose(filters=1,kernel_size=(11,11,1),strides=(4,4,1),padding='valid',a
  model.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])
  return model
```

Train model over training data to learn the video sequences

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
X_train = getTrainingData()
frames = X_train.shape[2]
#Need to make number of frames divisible by 10
frames = frames-frames%10
X_train = X_train[:,:,:frames]
X_train = X_train.reshape(-1,227,227,10)
X_train = np.expand_dims(X_train,axis=4)
```

```
print(X_train.shape)
Y_train = X_train.copy()
epochs = 100
#args.n_epochs
batch_size = 1
if __name__=="__main__":
  model=lo_model()
  #model.save('mymodel.h5')
  callback_save = ModelCheckpoint('AnomalyDetector.h5',monitor='val_loss', save_best_only=Tru
  callback_early_stopping = EarlyStopping(monitor='val_loss', patience=3)
  print('Model has been loaded')
  model.fit(X_train,Y_train,
        batch_size=batch_size,
        epochs=epochs,
        callbacks = [callback_save,callback_early_stopping],
        validation_split=0.2
)
  #batch_size=Number of samples per gradient update
```

```
455
504
496
504
272
504
367
339
464
408
261
49
122
170
118
82
5115
5115 227 227
(227, 227, 5115)
(511, 227, 227, 10, 1)
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793:

Model has been loaded
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/op
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

Train on 408 samples, validate on 103 samples
Epoch 1/100
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

408/408 [==============================] - 116s 285ms/step - loss: 0.1207 - acc: 0.6722
Epoch 2/100
```

```
408/408 [==============================] - 107s 263ms/step - loss: 0.0791 - acc: 0.7298
Epoch 3/100
408/408 [==============================] - 107s 263ms/step - loss: 0.0765 - acc: 0.7325
Epoch 4/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0754 - acc: 0.7335
Epoch 5/100
408/408 [==============================] - 107s 263ms/step - loss: 0.0749 - acc: 0.7339
Epoch 6/100
408/408 [==============================] - 107s 261ms/step - loss: 0.0747 - acc: 0.7341
Epoch 7/100
408/408 [==============================] - 107s 261ms/step - loss: 0.0744 - acc: 0.7343
Epoch 8/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0743 - acc: 0.7345
Epoch 9/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0742 - acc: 0.7345
Epoch 10/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0740 - acc: 0.7348
Epoch 11/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0651 - acc: 0.7408
Epoch 12/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0446 - acc: 0.7589
Epoch 13/100
408/408 [==============================] - 107s 261ms/step - loss: 0.0382 - acc: 0.7671
Epoch 14/100
408/408 [==============================] - 107s 261ms/step - loss: 0.0354 - acc: 0.7700
Epoch 15/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0334 - acc: 0.7719
Epoch 16/100
408/408 [==============================] - 107s 263ms/step - loss: 0.0322 - acc: 0.7730
Epoch 17/100
408/408 [==============================] - 108s 265ms/step - loss: 0.0313 - acc: 0.7739
Epoch 18/100
408/408 [==============================] - 108s 265ms/step - loss: 0.0305 - acc: 0.7746
Epoch 19/100
408/408 [==============================] - 109s 267ms/step - loss: 0.0300 - acc: 0.7751
Epoch 20/100
408/408 [==============================] - 109s 267ms/step - loss: 0.0295 - acc: 0.7755
Epoch 21/100
408/408 [==============================] - 108s 265ms/step - loss: 0.0291 - acc: 0.7759
Epoch 22/100
408/408 [==============================] - 108s 264ms/step - loss: 0.0287 - acc: 0.7763
Epoch 23/100
408/408 [==============================] - 107s 263ms/step - loss: 0.0284 - acc: 0.7765
Epoch 24/100
408/408 [==============================] - 108s 266ms/step - loss: 0.0281 - acc: 0.7767
Epoch 25/100
408/408 [==============================] - 110s 269ms/step - loss: 0.0278 - acc: 0.7770
Epoch 26/100
408/408 [==============================] - 109s 268ms/step - loss: 0.0276 - acc: 0.7772
Epoch 27/100
408/408 [==============================] - 108s 264ms/step - loss: 0.0274 - acc: 0.7774
Epoch 28/100
408/408 [==============================] - 108s 265ms/step - loss: 0.0272 - acc: 0.7775
Epoch 29/100
408/408 [==============================] - 107s 263ms/step - loss: 0.0270 - acc: 0.7777
Epoch 30/100
408/408 [==============================] - 108s 264ms/step - loss: 0.0269 - acc: 0.7778
Epoch 31/100
```

```
408/408 [==============================] - 108s 264ms/step - loss: 0.0267 - acc: 0.7779
Epoch 32/100
408/408 [==============================] - 108s 264ms/step - loss: 0.0265 - acc: 0.7781
Epoch 33/100
408/408 [==============================] - 108s 264ms/step - loss: 0.0264 - acc: 0.7782
Epoch 34/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0263 - acc: 0.7782
Epoch 35/100
408/408 [==============================] - 106s 261ms/step - loss: 0.0262 - acc: 0.7783
Epoch 36/100
408/408 [==============================] - 106s 259ms/step - loss: 0.0261 - acc: 0.7784
Epoch 37/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0260 - acc: 0.7785
Epoch 38/100
408/408 [==============================] - 106s 260ms/step - loss: 0.0259 - acc: 0.7786
Epoch 39/100
408/408 [==============================] - 106s 261ms/step - loss: 0.0258 - acc: 0.7786
Epoch 40/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0257 - acc: 0.7787
Epoch 41/100
408/408 [==============================] - 106s 261ms/step - loss: 0.0257 - acc: 0.7788
Epoch 42/100
408/408 [==============================] - 108s 265ms/step - loss: 0.0256 - acc: 0.7788
Epoch 43/100
408/408 [==============================] - 109s 266ms/step - loss: 0.0255 - acc: 0.7789
Epoch 44/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0254 - acc: 0.7790
Epoch 45/100
408/408 [==============================] - 107s 263ms/step - loss: 0.0254 - acc: 0.7791
Epoch 46/100
408/408 [==============================] - 107s 263ms/step - loss: 0.0253 - acc: 0.7793
Epoch 47/100
408/408 [==============================] - 107s 263ms/step - loss: 0.0252 - acc: 0.7795
Epoch 48/100
408/408 [==============================] - 107s 263ms/step - loss: 0.0250 - acc: 0.7798
Epoch 49/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0249 - acc: 0.7801
Epoch 50/100
408/408 [==============================] - 107s 262ms/step - loss: 0.0248 - acc: 0.7804
Epoch 51/100
408/408 [==============================] - 106s 261ms/step - loss: 0.0247 - acc: 0.7807
Epoch 52/100
408/408 [==============================] - 106s 260ms/step - loss: 0.0246 - acc: 0.7808
Epoch 53/100
408/408 [==============================] - 106s 259ms/step - loss: 0.0245 - acc: 0.7810
Epoch 54/100
408/408 [==============================] - 106s 261ms/step - loss: 0.0244 - acc: 0.7812
Epoch 55/100
408/408 [==============================] - 106s 260ms/step - loss: 0.0243 - acc: 0.7813
Epoch 56/100
408/408 [==============================] - 106s 259ms/step - loss: 0.0243 - acc: 0.7814
Epoch 57/100
408/408 [==============================] - 106s 260ms/step - loss: 0.0242 - acc: 0.7815
Epoch 58/100
408/408 [==============================] - 106s 259ms/step - loss: 0.0241 - acc: 0.7816
Epoch 59/100
408/408 [==============================] - 108s 265ms/step - loss: 0.0240 - acc: 0.7817
Epoch 60/100
```

```
    Epoch 60/100
    408/408 [==============================] - 106s 261ms/step - loss: 0.0240 - acc: 0.7818
    Epoch 61/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0239 - acc: 0.7819
    Epoch 62/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0239 - acc: 0.7820
    Epoch 63/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0238 - acc: 0.7821
    Epoch 64/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0238 - acc: 0.7821
    Epoch 65/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0237 - acc: 0.7822
    Epoch 66/100
    408/408 [==============================] - 106s 259ms/step - loss: 0.0236 - acc: 0.7823
    Epoch 67/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0236 - acc: 0.7824
    Epoch 68/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0235 - acc: 0.7825
    Epoch 69/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0235 - acc: 0.7826
    Epoch 70/100
    408/408 [==============================] - 105s 257ms/step - loss: 0.0234 - acc: 0.7826
    Epoch 71/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0233 - acc: 0.7827
    Epoch 72/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0233 - acc: 0.7828
    Epoch 73/100
    408/408 [==============================] - 106s 259ms/step - loss: 0.0232 - acc: 0.7829
    Epoch 74/100
    408/408 [==============================] - 106s 259ms/step - loss: 0.0232 - acc: 0.7830
    Epoch 75/100
    408/408 [==============================] - 107s 262ms/step - loss: 0.0231 - acc: 0.7830
    Epoch 76/100
    408/408 [==============================] - 107s 261ms/step - loss: 0.0231 - acc: 0.7831
    Epoch 77/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0230 - acc: 0.7832
    Epoch 78/100
    408/408 [==============================] - 106s 259ms/step - loss: 0.0230 - acc: 0.7833
    Epoch 79/100
    408/408 [==============================] - 106s 259ms/step - loss: 0.0229 - acc: 0.7834
    Epoch 80/100
    408/408 [==============================] - 105s 258ms/step - loss: 0.0228 - acc: 0.7834
    Epoch 81/100
```

Creating frames from testing video

```
    408/408 [==============================] - 105s 258ms/step - loss: 0.0227 - acc: 0.7836
```

```
def createTestingDataPerVideo( x ):
  try:
    # creating a folder named data
    if not os.path.exists('TestingData'+str(x)):
      os.makedirs('TestingData'+ str(x))
    # if not created then raise error
  except OSError:
    print ('Error: Creating directory of TestingData')
    # frame
  currentframe = 0
  count = 0
  count = 0
  videoPath = "Avenue Dataset/testing_videos/{}.avi".format(x)
  cam = cv2.VideoCapture(videoPath)
  while(True):
    # reading from frame
    cam.set(cv2.CAP_PROP_POS_MSEC,(currentframe*120))
    ret,frame = cam.read()
    if ret:
      print(ret)
      # if video is still left continue creating images
      name = './TestingData' + str(x) + '/frame' + str(count) + '.jpg'
      print ('Creating...' + name)
      # writing the extracted images
      cv2.imwrite(name, frame)
      # increasing counter so that it will
      # show how many frames are created
      currentframe += 1
      count += 1
    else:
      break
  # Release all space and windows once done
  cam.release()
  cv2.destroyAllWindows()
```

Create testing data

```
def createTestingData():
  for k in range(1,22):
    createTestingDataPerVideo(k)
createTestingData()
print('Testing data created')
```

Preprocessing of testing data

```
def funn( a, x ):
  teststore=[]
  framepath = a + str(x)
  images = os.listdir(framepath)
  print(len(images))
  for image in images:
    image_path=framepath+ '/'+ image
    img = cv2.imread(image_path)
    img = cv2.resize(img,(227,227))
    #Convert the Image to Grayscale
    gray = 0.2989*img[:,:,0] + 0.5870*img[:,:,1] + 0.1140*img[:,:,2]
    teststore.append(gray)
  teststore_array = np.array(teststore)
  a,b,c=teststore_array.shape
  #Reshape to (227,227,batch_size)
  teststore_array.resize(b,c,a)
  #Normalize
  teststore_array_normalize=(teststore_array-teststore_array.mean())/(teststore_array.std())
  #Clip negative Values
  teststore_clip=np.clip(teststore_array_normalize,0,1)
  return teststore_clip
```

Calculation of reconstruction error

```
def mean_squared_loss(x1,x2):
  ''' Compute Euclidean Distance Loss  between
  input frame and the reconstructed frame'''
  diff=x1-x2
  a,b,c,d,e=diff.shape
  n_samples=a*b*c*d*e
  sq_diff=diff**2
  Sum=sq_diff.sum()
  dist=np.sqrt(Sum)
  mean_dist=dist/n_samples
  return mean_dist
```

Testing done over both training and testing videos

```
trainVector = []
```

Testing over abnormal videos

```
from keras.models import load_model
for k in range(1, 22):
  threshold=0.00046
  model = load_model('AnomalyDetector.h5')
  X_test = funn('TestingData', k)
  frames = X_test.shape[2]
  #Need to make number of frames divisible by 10
  flag = 0 #Overall video flag
  frames = frames-frames%10
  X_test = X_test[:,:,:frames]
  X_test = X_test.reshape(-1,227,227,10)
  X_test = np.expand_dims(X_test,axis=4)
  for number,bunch in enumerate(X_test):
    n_bunch=np.expand_dims(bunch,axis=0)
    reconstructed_bunch=model.predict(n_bunch)
    loss=mean_squared_loss(n_bunch,reconstructed_bunch)
    if loss>threshold:
```

```
    if loss>threshold:
      #print("Anomalous bunch of frames at bunch number {} = {}".format(number,loss))
      flag=1
    #else:
      #print("Non Anomalous bunch of frames at bunch number {} = {}".format(number,loss))
  if flag == 1:
    print('Testing {} has anomalous activity'.format(k))
    trainVector.append(1)
  else:
    trainVector.append(0)
```

Testing over normal videos

```
for k in range(1, 17):
  threshold=0.00046
  model = load_model('AnomalyDetector.h5')
  X_test = funn('TrainingData', k)
  frames = X_test.shape[2]
  #Need to make number of frames divisible by 10
  flag = 0 #Overall video flag
  frames = frames-frames  4)
  for number,bunch in enumerate(X_test):
    n_bunch=np.expand_dims(bunch,axis=0)
    reconstructed_bunch=model.predict(n_bunch)
    loss=mean_squared_loss(n_bunch,reconstructed_bunch)
    if loss>threshold:
      #print("Anomalous bunch of frames at bunch number {} = {}".format(number,loss))
      flag=1
    #else:
      #print("Non Anomalous bunch of frames at bunch number {} = {}".format(number,loss))
  if flag == 1:
    print('Training {} has anomalous activity'.format(k))
    trainVector.append(1)
  else:
    trainVector.append(0)

"""pr_auc = metrics.auc(recall, precision)

plt.title("Precision-Recall vs Threshold Chart")
plt.plot(thresholds, precision[: -1], "b--", label="Precision")
plt.plot(thresholds, recall[: -1], "r--", label="Recall")
plt.ylabel("Precision, Recall")
plt.xlabel("Threshold")
plt.legend(loc="lower left")
plt.ylim([0,1])"""
```

```
trainVector=[1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,0,1,0,0,0,0,0,0,1,1,0,1,0,1,1,0,1,1]
```

Accuracy calculation by construction of confusion matrix

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
testVector = []
for i in range(1,22):
  testVector.append(1)
for i in range (1,17):
  testVector.append(0)
results = confusion_matrix(testVector, trainVector)
print('Confusion Matrix :')
print(results)
print('Accuracy Score :',accuracy_score(testVector, trainVector)) # testtVector is y_true tra
print('Report : ')
print(classification_report(testVector, trainVector))
```

Regularity score calculation

```
from keras.models import load_model
def plotRegularityScore(data, x):
  model = load_model('AnomalyDetector.h5')
  sr = []
  sa = []
  X_test = funn(data, x)
  frames = X_test.shape[2]
  #Need to make number of frames divisible by 10
  flag = 0 #Overall video flag
  frames = frames-frames%10
  X_test = X_test[:,:,:frames]
  X_test = X_test.reshape(-1,227,227,10)
  X_test = np.expand_dims(X_test,axis=4)
  for number,bunch in enumerate(X_test):
    n_bunch=np.expand_dims(bunch,axis=0)
    reconstructed_bunch=model.predict(n_bunch)
    loss=mean_squared_loss(n_bunch,reconstructed_bunch)
    sa.append(loss)
  sa = (sa - np.min(sa)) / np.max(sa)
  sr = 1.0 - sa
  plt.plot(sr)
  plt.ylabel('Regularity Score Sr(t)')
  plt.xlabel('Frame Number(t)')
  plt.xticks(np.arange(0,30, 5))
  plt.yticks(np.arange(0.6, 1, 0.05))
  plt.show()
```

```
plotRegularityScore('TestingData',1)
```

```
plotRegularityScore('TrainingData', 1)
plotRegularityScore('TestingData', 4)
```

```
plotRegularityScore('TestingData', 7)
```

```
plotRegularityScore('TrainingData', 3)
plotRegularityScore('TestingData', 14)
```

```
plotRegularityScore('TestingData', 15)
plotRegularityScore('TestingData', 16)
```

```
from keras import models
from keras.models import load_model
model = load_model('AnomalyDetector.h5')
layer_outputs = [layer.output for layer in model.layers[:7]] # Extracts the outputs of the to
activation_model = models.Model(inputs=model.input, outputs=layer_outputs) # Creates a model
```

```
X = funn('TrainingData', 1)
frames = X.shape[2]
#Need to make number of frames divisible by 10
flag = 0 #Overall video flag
frames = frames-frames%10
X = X[:,:,:frames]
X = X.reshape(-1,227,227,10)
X = np.expand_dims(X,axis=4)
```

```
model.summary()
```

```
activations = activation_model.predict(X) # Returns a list of five Numpy arrays: one array pe
```

Activation units visualization

Input sequence frame

```
plt.matshow(X[0,:,:,0,0], cmap='viridis')
```

Activations of different layers

First Layer activation unit

```
first_layer_activation = activations[0]
print(first_layer_activation.shape)
plt.matshow(first_layer_activation[0, :, :,0,0], cmap='viridis')
```

Second layer activation unit

```
second_layer_activation = activations[1]
print(second_layer_activation.shape)
plt.matshow(second_layer_activation[0, :, :,0,0], cmap='viridis')
```

Third layer activation unit

```
third_layer_activation = activations[2]
print(third_layer_activation.shape)
plt.matshow(third_layer_activation[0, :, :,0,0], cmap='viridis')
```

Fourth layer activation unit

```
fourth_layer_activation = activations[3]
print(fourth_layer_activation.shape)
plt.matshow(fourth_layer_activation[0, :, :,0,0], cmap='viridis')
```

Fifth layer activation units

```
fifth_layer_activation = activations[4]
print(fifth_layer_activation.shape)
plt.matshow(fifth_layer_activation[0, :, :,0,0], cmap='viridis')
```

Sixth layer activation units

```
sixth_layer_activation = activations[5]
print(sixth_layer_activation.shape)
plt.matshow(sixth_layer_activation[0, :, :,0,0], cmap='viridis')
```

Final layer activation units

```
final_layer_activation = activations[6]
print(final_layer_activation.shape)
plt.matshow(final_layer_activation[0, :, :,0,0], cmap='viridis')
```

Features visualization

Features of conv layer

Features of first convolution layer

```
filters, biases = model.layers[0].get_weights()
# normalize filter values to 0-1 so we can visualize them
f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)
# plot first few filters
n_filters, ix = 6, 1
f = filters[:, :, 0,0, 0]
#ax = plt.subplot(n_filters, 3, ix)
#ax.set_xticks([])
#ax.set_yticks([])
# plot filter channel in grayscale
plt.imshow(f, cmap='gray')
# show the figure
plt.show()
```

```
filters, biases = model.layers[0].get_weights()
# normalize filter values to 0-1 so we can visualize them
f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)
# plot first few filters
n_filters, ix = 6, 1
f = filters[:, :, 0,0, 1]
#ax.set_xticks([])
#ax.set_yticks([])
# plot filter channel in grayscale
# plot filter channel in grayscale
plt.imshow(f, cmap='gray')
# show the figure
plt.show()
```

Features of second convolution layer

```
filters, biases = model.layers[1].get_weights()
# normalize filter values to 0-1 so we can visualize them
f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)
# plot first few filters
n_filters, ix = 6, 1
f = filters[:, :, 0,0, 0]
#ax.set_xticks([])
#ax.set_yticks([])
# plot filter channel in grayscale
plt.imshow(f, cmap='gray')
# show the figure
plt.show()
```