CSYE 7374 – Cognitive Computing & Neural Networks
Unsupervised Abnormal Event Detection

Gaurang Davda
NUID 001826203

1. Abstract

For Anomaly detection in videos, Instead of treating it as supervised learning and Labeling the videos as Normal and abnormal, another approach is used and the anomaly detection is done by unsupervised learning. It is difficult to obtain abnormal videos as compared to normal videos. Even in self automated cars, the biggest challenge is to get accident videos because it is very difficult to get them and also to generate them. Spatiotemporal architecture is proposed for anomaly detection in videos including crowded scenes. It contains two main components, one for spatial feature representation, and the other for learning the temporal evolution of the spatial features.Model is trained by only normal videos(some outliers) and while testing, when abnormal videos are given to this trained model, the reconstruction error for such videos goes above threshold and anomaly is detected.

2. Keywords:

Unsupervised, Spatiotemporal, Autoencoder, Long Short Term Memory (LSTM), Convolutional LSTM, Anomaly, Video

3. Introduction

Suspicious events that are of interest in long video sequences, such as surveillance footage usually have an extremely low probability of occurring. Manually detecting such events, or anomalies, is a very meticulous job that often requires more manpower than is generally available. Hence, there is a need for Automate detection. Treating the task as a binary classification problem (normal and abnormal) proved it being effective and accurate, but the practicality of such method is limited since footages of abnormal events are difficult to obtain due to its rarity. Hence, it is more efficient to train a model using little to no supervision, including spatiotemporal features and autoencoders . Unlike supervised methods, these methods only require unlabelled video footages which contain little or no abnormal event, which are easy to obtain in real-world applications.

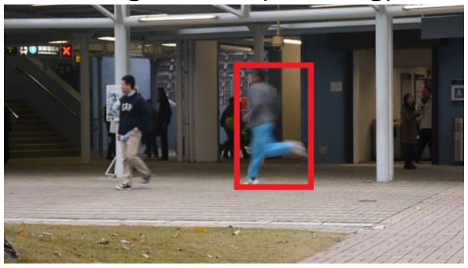
In this project, video data set is represented by a set of general features, which are inferred automatically from a long video footage through a deep learning approach. Specifically, a deep neural network composed of a stack of convolutional autoencoders was used to process video frames in an unsupervised manner that captured spatial structures in the data, which, grouped together, compose the video representation. Then, this representation is fed into a stack of convolutional temporal autoencoders to learn the regular temporal patterns. The method described here is based on the principle that when an abnormal event occurs, the most recent frames of video will be significantly different than the older frames. Trained an end-to-end model that consists of a spatial feature extractor and a temporal encoder-decoder which together learns the temporal patterns of the input volume of frames. The model is trained with video volumes consists of only normal scenes, with the objective to minimize the reconstruction error between the input video volume and the output video volume reconstructed by the learned model. After the model is properly trained, normal video volume is expected to have low reconstruction error, whereas video volume consisting of abnormal scenes is expected to have high reconstruction error. By thresholding on the error produced by each testing input volumes, our system will be able to detect when an abnormal event occurs. The model architecture was inspired from Reference paper [3] which was implemented in MATLAB, But this was considered as the base model and many improvements were made over this model by doing Hyperparameter Tuning.

4. Methods

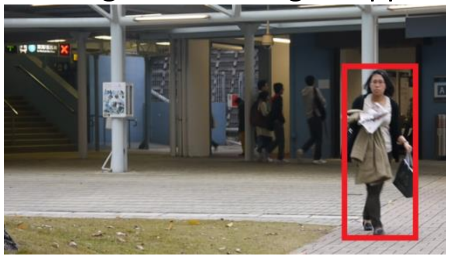
Dataset: Avenue Dataset is downloaded from http://www.cse.cuhk.edu.hk/leojia/projects/detectabnormal/dataset.html. Avenue Dataset contains 16 training and 21 testing video clips. The videos are captured in CUHK campus avenue with 30652 (15328 training, 15324 testing) frames in total. This dataset accompanies paper "Abnormal Event Detection at 150 FPS in Matlab, Cewu Lu, Jianping Shi, Jiaya Jia, International Conference on Computer Vision, (ICCV), 2013".

Video Description: The training videos capture normal situations. Testing videos include both normal and abnormal events. Three abnormal samples are shown as follows.

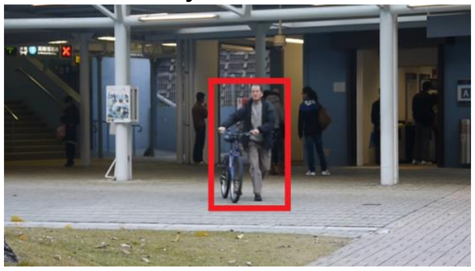
Strange Action(Running)



Walking in the wrong or opposite direction



Abnormal object



Preprocessing:

The task of this stage is to convert raw data to the aligned and acceptable input for the model. Each frame is extracted from the raw videos and resized to 227 × 227. To ensure that the input images are all on the same scale, the pixel values are scaled between 0 and 1 and subtracted every frame from its global mean image for normalization. The mean image is calculated by averaging the pixel values at each location of every frame in the training dataset. After that, the images are converted to grayscale to reduce dimensionality. The processed images are then normalized to have zero mean and unit variance. The input to the model is video volumes, where each volume consists of 10 consecutive frames with various skipping strides. As the number of parameters in this model is large, large amount of training data is needed. Data augmentation is performed in the temporal dimension to increase the size of the training dataset. To generate these volumes, frames are concatenated with stride-1, stride-2, and stride-3. For example, the first stride-1 sequence is made up of frame {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, whereas the first stride-2 sequence contains frame number {1, 3, 5, 7, 9, 11, 13, 15, 17, 19}, and stride-3 sequence would contain frame number {1, 4, 7, 10, 13, 16, 19, 22, 25, 28}. Now the input is ready for model training.

Feature Learning:
Convolutional spatiotemporal autoencoder is used to learn the regular patterns in the training videos. Architecture consists of two parts — spatial autoencoder for learning spatial structures of each video frame, and temporal encoder-decoder for learning temporal patterns of the encoded spatial structures. As illustrated in Figure 1 and 2, the spatial encoder and decoder have two convolutional and deconvolutional layers respectively, while the temporal encoder is a three-layer convolutional long short term memory (LSTM) model. Convolutional layers are well-known for its superb performance in object recognition, while LSTM model is widely used for sequence learning and time-series modelling.

Autoencoder:
Autoencoders consist of two stages: encoding and decoding. It was first used to reduce dimensionality by setting the number of encoder output units less than the input. The model is usually trained using back-propagation in an unsupervised manner, by minimizing the reconstruction error of the decoding results from the original inputs. With the activation function chosen to be nonlinear, an autoencoder can extract more useful features than some common linear transformation methods such as PCA.

Spatial Convolution:
The primary purpose of convolution in case of a convolutional network is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. Mathematically, convolution operation performs dot products between the filters and local regions of the input. Suppose that we have some $n \times n$ square input layer which is followed by the convolutional layer. If we use an $m \times m$ filter W, the convolutional layer output will be of size $(n-m+1)\times(n-m+1)$. A convolutional network learns the values of these filters on its own during the training process, although we still need to specify parameters such as the number of filters, filter size, the number of layers before training. With more number of filters we have, more image features get extracted and the better the network becomes at recognizing patterns in unseen images. However, more filters would add to computational time and exhaust memory faster, so we need to find balance by not setting the number of filters too large.

Figure 1: Proposed network architecture. It takes a sequence of length T as input, and output a reconstruction of the input sequence. The numbers at the rightmost denote the output size of each layer. The spatial encoder takes a batch of 10 frames, the encoded features of 10 frames are concatenated and fed into temporal encoder for

motion encoding. The decoders mirror the encoders to reconstruct the video volume.

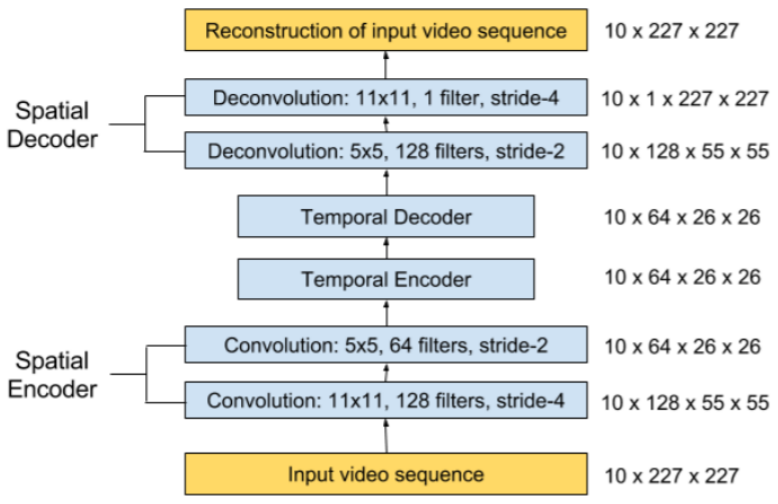
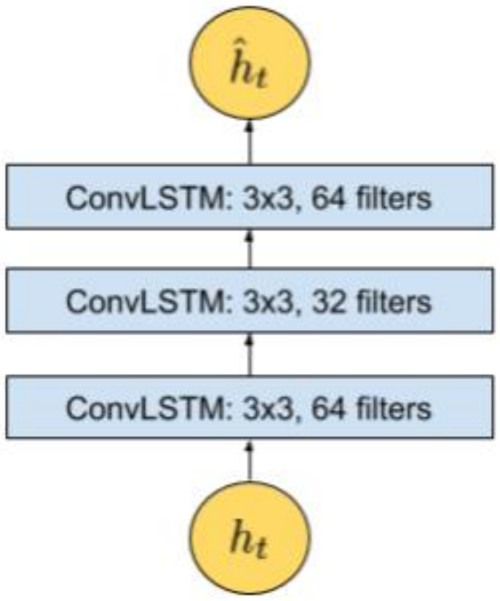


Figure 2: The zoomed-in architecture at time t, where t is the input vector at this time step. The temporal encoder-decoder model has 3 convolutional LSTM (ConvLSTM) layers


$\hat{h}_t$
ConvLSTM: 3x3, 64 filters
ConvLSTM: 3x3, 32 filters
ConvLSTM: 3x3, 64 filters
h_t

RNNs can make use of information in arbitrarily long sequences, but in practice, they are limited to looking back only a few steps due to vanishing gradients.

To overcome this problem, a variant of RNN is introduced: long short term memory (LSTM) model which incorporates a recurrent gate called forget gate. With the new structure, LSTMs prevent backpropagated errors from vanishing or exploding, thus can work on long sequences and they can be stacked together to capture higher level information. The formulation of a typical LSTM unit is summarized with Figure 3 and equations (1) through (6).

$$f_t = \sigma(W_f * [h_{t-1}, x_t, C_{t-1}] + b_f) \tag{7}$$

$$i_t = \sigma(W_i * [h_{t-1}, x_t, C_{t-1}] + b_i) \tag{8}$$

$$\hat{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C) \tag{9}$$

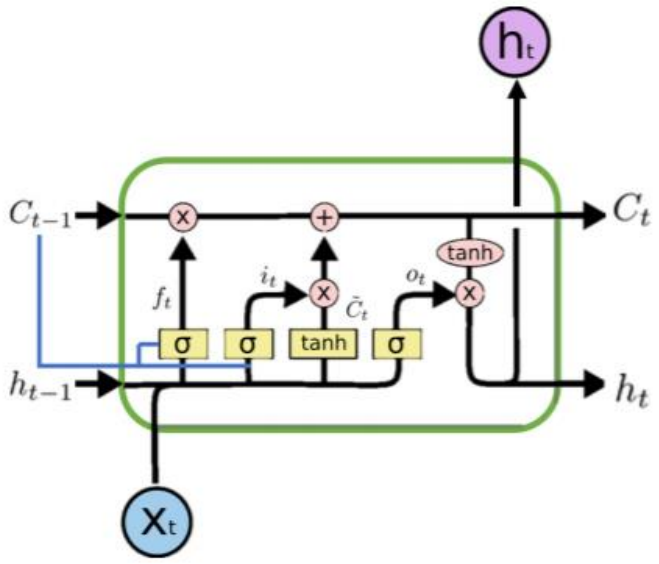
$$C_t = f_t \otimes C_{t-1} + i_t \otimes \hat{C}_t \tag{10}$$

$$o_t = \sigma(W_o * [h_{t-1}, x_t, C_{t-1}] + b_o) \tag{11}$$

$$h_t = o_t \otimes tanh(C_t) \tag{12}$$

Equation (1) represents the forget layer, (2) and (3) are where new information is added, (4) combines old and new information, whereas (5) and (6) output what has been learned so far to the LSTM unit at the next timestep. The variable xt denotes the input vector, ht denotes the hidden state, and Ct denotes the cell state at time t. W are the trainable weight matrices, b are the bias vectors, and the symbol ⊗ denotes the Hadamard product.

Convolutional LSTM:



Convolutional LSTM, a variant of the LSTM architecture, namely Convolutional Long

Short-term Memory (ConvLSTM) model was introduced by Shi et al. in [1](1st Research paper in reference) and has been recently utilized by Patraucean et al. in [2](second Research paper in reference) for video frame prediction. Compared to the usual fully connected LSTM (FC-LSTM), ConvLSTM has its matrix operations replaced with convolutions. By using convolution for both input-to-hidden and hidden-to-hidden connections, ConvLSTM requires fewer weights and yield better spatial feature maps. The formulation of the ConvLSTM unit can be summarized with (7) through (12).

$$f_t = \sigma(W_f * [h_{t-1}, x_t, C_{t-1}] + b_f) \tag{7}$$

$$i_t = \sigma(W_i * [h_{t-1}, x_t, C_{t-1}] + b_i) \tag{8}$$

$$\hat{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C) \tag{9}$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \hat{C}_t \tag{10}$$

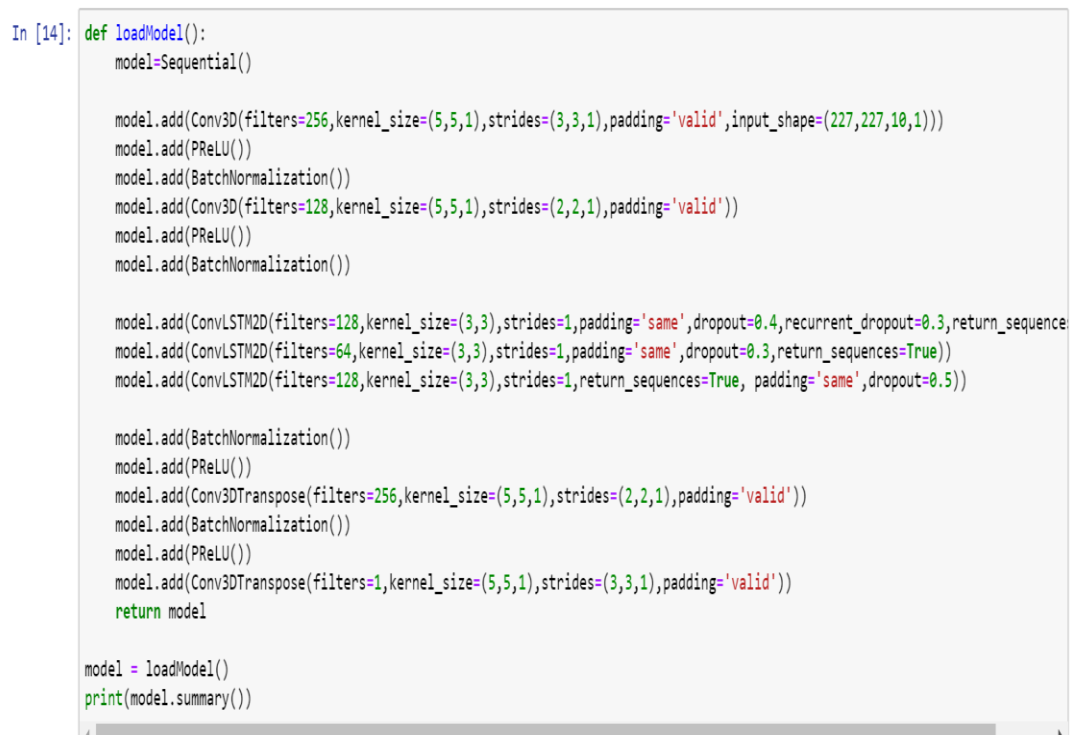
$$o_t = \sigma(W_o * [h_{t-1}, x_t, C_{t-1}] + b_o) \tag{11}$$

$$h_t = o_t \otimes tanh(C_t) \tag{12}$$

While the equations are similar in nature to (1) through (6), the input is fed in as images, while the set of weights for every connection is replaced by convolutional filters (the symbol $*$ denotes a convolution operation). This allows ConvLSTM work better with images than the FC-LSTM due to its ability to propagate spatial characteristics temporally through each ConvLSTM state. Note that this convolutional variant also adds an optional ’peephole’ connections to allow the unit to derive past information better.

Model:

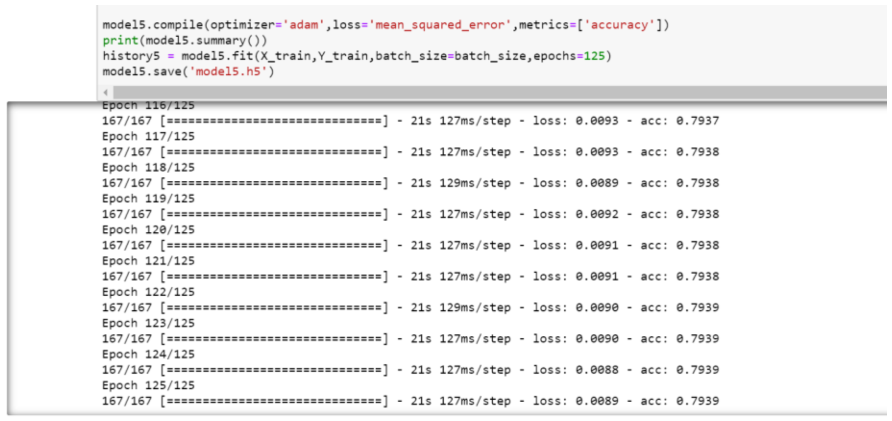
After experimenting with different models by Hyperparamter Tuning, highest accuracy is achieved by the below model



```
In [14]: def loadModel():
             model=Sequential()

             model.add(Conv3D(filters=256,kernel_size=(5,5,1),strides=(3,3,1),padding='valid',input_shape=(227,227,10,1)))
             model.add(PReLU())
             model.add(BatchNormalization())
             model.add(Conv3D(filters=128,kernel_size=(5,5,1),strides=(2,2,1),padding='valid'))
             model.add(PReLU())
             model.add(BatchNormalization())

             model.add(ConvLSTM2D(filters=128,kernel_size=(3,3),strides=1,padding='same',dropout=0.4,recurrent_dropout=0.3,return_sequence
             model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,padding='same',dropout=0.3,return_sequences=True))
             model.add(ConvLSTM2D(filters=128,kernel_size=(3,3),strides=1,return_sequences=True, padding='same',dropout=0.5))

             model.add(BatchNormalization())
             model.add(PReLU())
             model.add(Conv3DTranspose(filters=256,kernel_size=(5,5,1),strides=(2,2,1),padding='valid'))
             model.add(BatchNormalization())
             model.add(PReLU())
             model.add(Conv3DTranspose(filters=1,kernel_size=(5,5,1),strides=(3,3,1),padding='valid'))
             return model

         model = loadModel()
         print(model.summary())
```

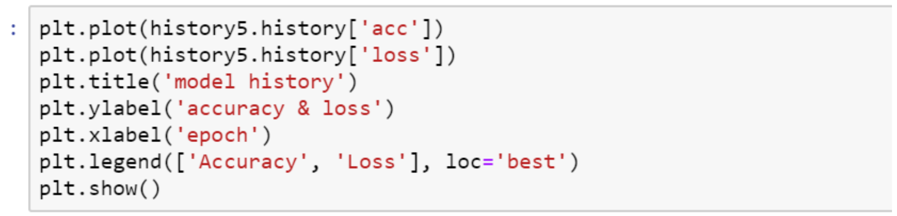
10 10 frames are passed to the model at once so that the model can find features in the sequence as described earlier.

Training accuracy of 79.39% was achieved in 125 epochs

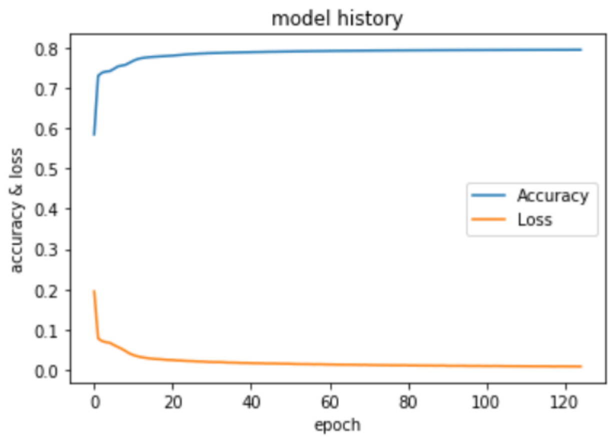


```
model5.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])
print(model5.summary())
history5 = model5.fit(X_train,Y_train,batch_size=batch_size,epochs=125)
model5.save('model5.h5')
```

```
Epoch 116/125
167/167 [==============================] - 21s 127ms/step - loss: 0.0093 - acc: 0.7937
Epoch 117/125
167/167 [==============================] - 21s 127ms/step - loss: 0.0093 - acc: 0.7938
Epoch 118/125
167/167 [==============================] - 21s 129ms/step - loss: 0.0089 - acc: 0.7938
Epoch 119/125
167/167 [==============================] - 21s 127ms/step - loss: 0.0092 - acc: 0.7938
Epoch 120/125
167/167 [==============================] - 21s 127ms/step - loss: 0.0091 - acc: 0.7938
Epoch 121/125
167/167 [==============================] - 21s 127ms/step - loss: 0.0091 - acc: 0.7938
Epoch 122/125
167/167 [==============================] - 21s 129ms/step - loss: 0.0090 - acc: 0.7939
Epoch 123/125
167/167 [==============================] - 21s 127ms/step - loss: 0.0090 - acc: 0.7939
Epoch 124/125
167/167 [==============================] - 21s 127ms/step - loss: 0.0088 - acc: 0.7939
Epoch 125/125
167/167 [==============================] - 21s 127ms/step - loss: 0.0089 - acc: 0.7939
```



```
plt.plot(history5.history['acc'])
plt.plot(history5.history['loss'])
plt.title('model history')
plt.ylabel('accuracy & loss')
plt.xlabel('epoch')
plt.legend(['Accuracy', 'Loss'], loc='best')
plt.show()
```



Evaluation :

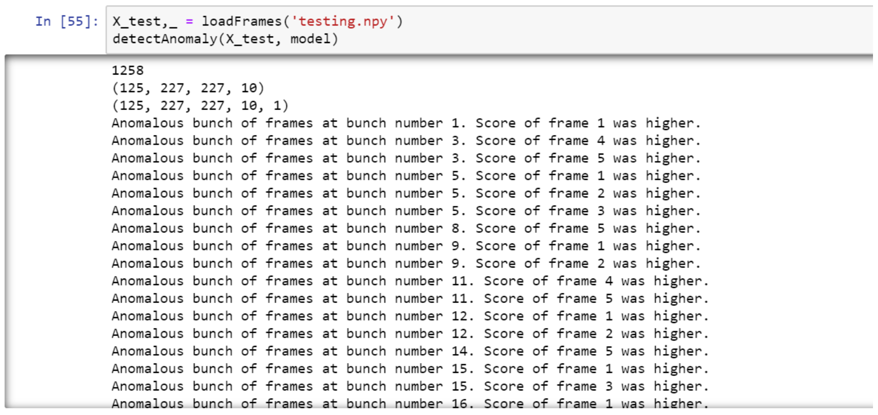
The reconstruction error of all pixel values I in frame t of the video sequence is taken as the Euclidean distance between the input frame and the reconstructed frame:

$$e(t) = ||x(t) - f_W(x(t))||_2$$

where fw is the learned weights by the spatiotemporal model
The reconstruction error of each frame determines whether the frame is classified as anomalous. The threshold determines how sensitive we wish the detection system to behave — for example, setting a low threshold makes the system become sensitive to the happenings in the scene, where more alarms would be triggered
The anomalous video will have a reconstruction error above threshold and in this way anomaly will be detected.

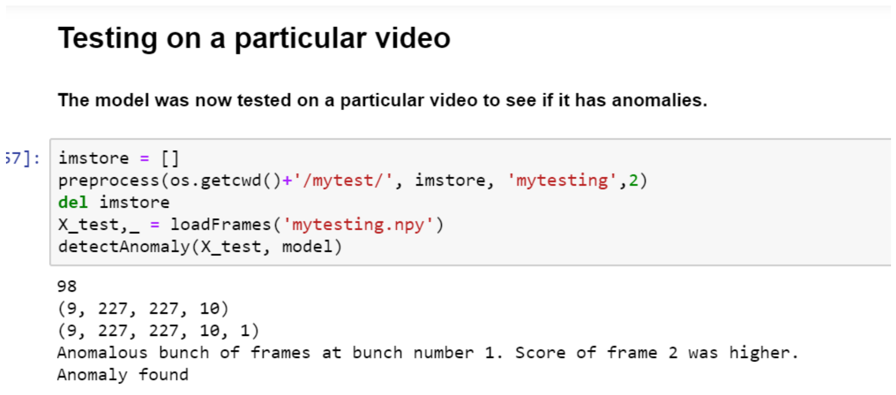
Results:

The model was first tested on the testing dataset.



```
In [55]: X_test,_ = loadFrames('testing.npy')
         detectAnomaly(X_test, model)
```

```
1258
(125, 227, 227, 10)
(125, 227, 227, 10, 1)
Anomalous bunch of frames at bunch number 1. Score of frame 1 was higher.
Anomalous bunch of frames at bunch number 3. Score of frame 4 was higher.
Anomalous bunch of frames at bunch number 3. Score of frame 5 was higher.
Anomalous bunch of frames at bunch number 5. Score of frame 1 was higher.
Anomalous bunch of frames at bunch number 5. Score of frame 2 was higher.
Anomalous bunch of frames at bunch number 5. Score of frame 3 was higher.
Anomalous bunch of frames at bunch number 8. Score of frame 5 was higher.
Anomalous bunch of frames at bunch number 9. Score of frame 1 was higher.
Anomalous bunch of frames at bunch number 9. Score of frame 2 was higher.
Anomalous bunch of frames at bunch number 11. Score of frame 4 was higher.
Anomalous bunch of frames at bunch number 11. Score of frame 5 was higher.
Anomalous bunch of frames at bunch number 12. Score of frame 1 was higher.
Anomalous bunch of frames at bunch number 12. Score of frame 2 was higher.
Anomalous bunch of frames at bunch number 14. Score of frame 5 was higher.
Anomalous bunch of frames at bunch number 15. Score of frame 1 was higher.
Anomalous bunch of frames at bunch number 15. Score of frame 3 was higher.
Anomalous bunch of frames at bunch number 16. Score of frame 1 was higher.
```



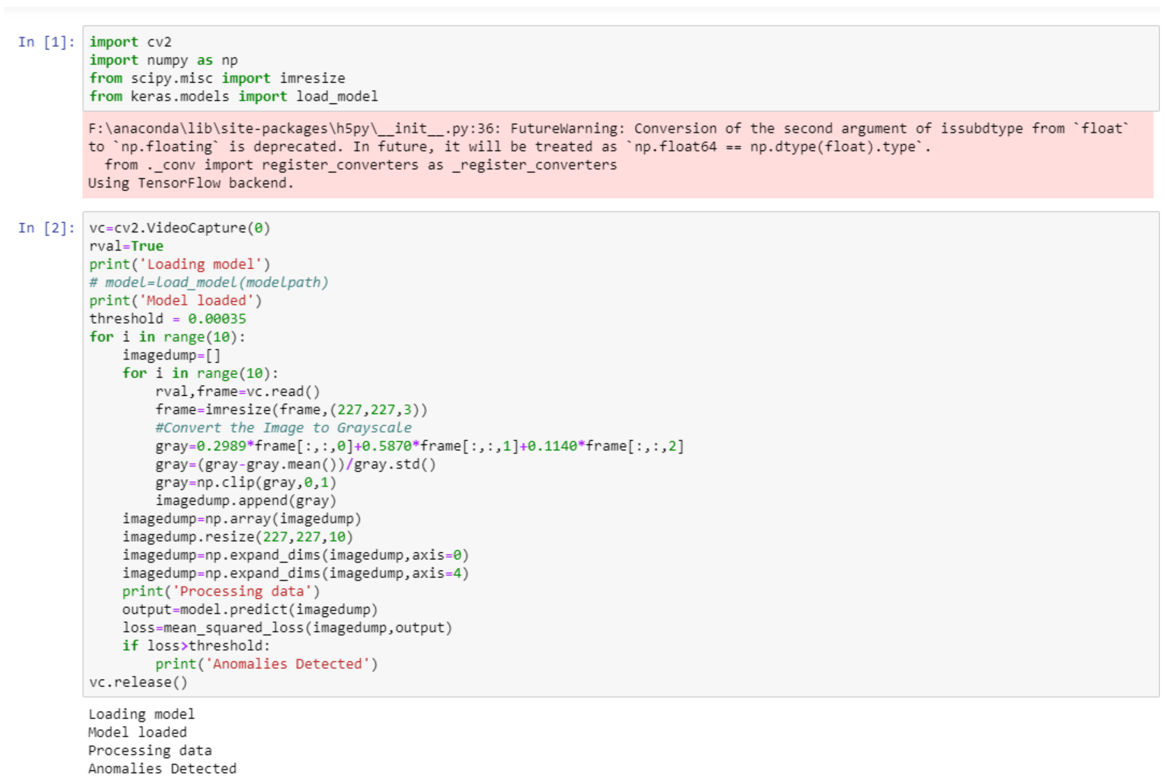
Testing on a particular video

The model was now tested on a particular video to see if it has anomalies.

```
57]: imstore = []
     preprocess(os.getcwd()+'/mytest/', imstore, 'mytesting',2)
     del imstore
     X_test,_ = loadFrames('mytesting.npy')
     detectAnomaly(X_test, model)
```

```
98
(9, 227, 227, 10)
(9, 227, 227, 10, 1)
Anomalous bunch of frames at bunch number 1. Score of frame 2 was higher.
Anomaly found
```

Testing on live feed
Live camera feed is captured and checked for anomalies if any.



```
In [1]: import cv2
        import numpy as np
        from scipy.misc import imresize
        from keras.models import load_model
```

```
F:\anaconda\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float`
to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
In [2]: vc=cv2.VideoCapture(0)
        rval=True
        print('Loading model')
        # model=load_model(modelpath)
        print('Model loaded')
        threshold = 0.00035
        for i in range(10):
            imagedump=[]
            for i in range(10):
                rval,frame=vc.read()
                frame=imresize(frame,(227,227,3))
                #Convert the Image to Grayscale
                gray=0.2989*frame[:,:,0]+0.5870*frame[:,:,1]+0.1140*frame[:,:,2]
                gray=(gray-gray.mean())/gray.std()
                gray=np.clip(gray,0,1)
                imagedump.append(gray)
            imagedump=np.array(imagedump)
            imagedump.resize(227,227,10)
            imagedump=np.expand_dims(imagedump,axis=0)
            imagedump=np.expand_dims(imagedump,axis=4)
            print('Processing data')
            output=model.predict(imagedump)
            loss=mean_squared_loss(imagedump,output)
            if loss>threshold:
                print('Anomalies Detected')
        vc.release()
```

```
Loading model
Model loaded
Processing data
Anomalies Detected
```

Conclusion:

In this Research, deep learning was applied to the challenging video anomaly detection problem. Anomaly detection is formulated as a spatiotemporal sequence outlier detection problem and applied a combination of spatial feature extractor and temporal sequencer ConvLSTM to tackle the problem. The ConvLSTM layer not only preserves the advantages of FC-LSTM but is also suitable for spatiotemporal data due to its inherent convolutional structure. By incorporating convolutional feature extractor in both spatial and temporal space into the encoding-decoding structure, an end-to-end trainable model is built for video anomaly detection. The advantage of my model is that it is semi-supervised – the only ingredient required is a long video segment containing only normal events in a fixed view.

Future Work:

Despite the models ability to detect abnormal events and its robustness to noise, depending on the activity complexity in the scene, it may produce more false alarms compared to other methods. For future work, on how to improve the result of video anomaly detection by active learning – having human feedback to update the learned model for better detection and reduced false alarms. One idea is to add a supervised module to the current system, which the supervised module works only on the video segments filtered by proposed method, then train a discriminative model to classify anomalies when enough video data has been acquired.

References:

Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.k., Woo, W.c.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: Proceedings of the 28th International Conference on Neural Information Processing Systems. pp. 802–810. NIPS'15, MIT Press, Cambridge, MA, USA (2015), http://dl.acm.org/citation.cfm?id=2969239.2969329

Patraucean, V., Handa, A., Cipolla, R.: Spatio-temporal video autoencoder with differentiable memory. International Conference On Learning Representations (2015), 1–10 (2016),

Yong Shean Chong, Yong Haur Tay: Abnormal Event Detection in Videos using Spatiotemporal Autoencoder https://arxiv.org/pdf/1701.01546.pdf
https://www.semanticscholar.org/paper/An-Overview-of-Deep-Learning-Based-Methods-for-and-Kiran-Thomas/7198f45e979d4e7bb2ad2f8a5f098ab196c532b6

https://www.semanticscholar.org/paper/An-Overview-of-Deep-Learning-Based-Methods-for-and-Kiran-Thomas/7198f45e979d4e7bb2ad2f8a5f098ab196c532b6

https://www.semanticscholar.org/paper/Improved-anomaly-detection-in-surveillance-videos-a-Khaleghi-Moin/1a5c917ec7763c2ff9619e6f19d02d2f254d236a

https://www.semanticscholar.org/paper/A-Short-Review-of-Deep-Learning-Methods-for-Group-Borja-Borja-Saval-Calvo/d9db8a4ce5ae4c4d03a55c648f4e7006838b6952

https://www.semanticscholar.org/paper/Context-encoding-Variational-Autoencoder-for-Zimmerer-Kohl/2f3a2e24fb0ea3a9b6a4ebf0430886fdfa3efdd3

https://machinelearningmastery.com/cnn-long-short-term-memory-networks/

https://arxiv.org/abs/1411.4389

https://www.coursera.org/lecture/nlp-sequence-models/long-short-term-memory-lstm-KXoay

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

https://www.ncbi.nlm.nih.gov/pubmed/22392705

https://arxiv.org/abs/1604.04574

https://www.researchgate.net/publication/221361667_Anomaly_detection_in_extremely_crowded_scenes_using_spatio-temporal_motion_pattern_models

https://pennstate.pure.elsevier.com/en/publications/adaptive-sparse-representations-for-video-anomaly-detection