MINOR PROJECT REPORT

5th Sem, 2020-2021

Jaypee Institute of Information Technology, Noida

TEAM MEMBERS:

Batch: B8

ADIT GOYAL (18103242)
AMRITANSHU AGRAWAL (18103243)
SHREYANSH DAS (18103263)

Students' Self Declaration for Open Source libraries and other source code usage in Minor Project

I / We, **Adit Goyal, Amritanshu Agrawal, and Shreyansh Das,** hereby declare the following usage of the open source code and prebuilt libraries in our minor project in **5th** Semester with the consent of our supervisor. We also measure the similarity percentage of pre written source code and our source code and the same is mentioned in the subsequent sections. This measurement is true to the best of our knowledge and abilities.

- Percentage of pre-written source code: 35%

Student ID	Student Name	Student signature
18103242	Adit Goyal	Agyl
18103243	Amritanshu Agrawal	Amritanshu
18103263	Shreyansh Das	Shreyansh

Declaration by Supervisor (To be filled by Supervisor only)

I, Dr. Manish Kumar Thakur declares that the project submitted above, titled "Unusual/Suspicious Activity Detection and Reporting/ CCTV Surveillance System", was conducted under my supervision. The project is original, and neither the project was copied from external sources, nor it was submitted earlier in JIIT. I authenticate this project.

(Any Remarks by Supervisor)

Dr. Manish Kumar Thakur

Signature (Supervisor)

TOPIC: Unusual/Suspicious Activity Detection and Reporting/ CCTV Surveillance System

Our project aims at easing the tedious task of going through hours of CCTV footage on a daily basis to find if something unusual happened. This application will be very useful for monitoring and security purposes in housing societies, office/commercial spaces and hostels/PGs.

Wardens in hostels won't have to go monitor the footage continuously to check for misbehaving students. Our application can be connected directly to the live CCTV footage and detect any unusual/suspicious activities that took place and report it to the administrator. The respective authority can review the details of such events and take the necessary action. The details that can be included in the report are: an alert that some event is flagged, the time it happened; and some still frames/screenshots of the suspicious activity.

Now, the wardens of hostels/security guards can simply review those specific activities during the time frame mentioned in the report and act upon it. This will save a ton of precious time and resources.

SYSTEM REQUIREMENTS

- **Software Requirements**
  - Operating system: Windows, Linux
  - Python
  - Python Libraries and features:
    - os: path, makedirs
    - numpy: array, hstack, round, random
    - pandas: DataFrame
    - matplotlib: pyplot, image
    - datetime
    - tensorflow: placeholder, train, nn, summary, global_variables_initializer, Session, layers, matmul
    - cv2: VideoCapture, imwrite, VideoWriter
    - sys
    - glob
    - sklearn: metrics
    - pickle
    - imageio
    - tkinter: filedialog, ttk
    - PIL: Image, ImageTK
    - smtplib: SMTP
    - base64
    - email: message, mime

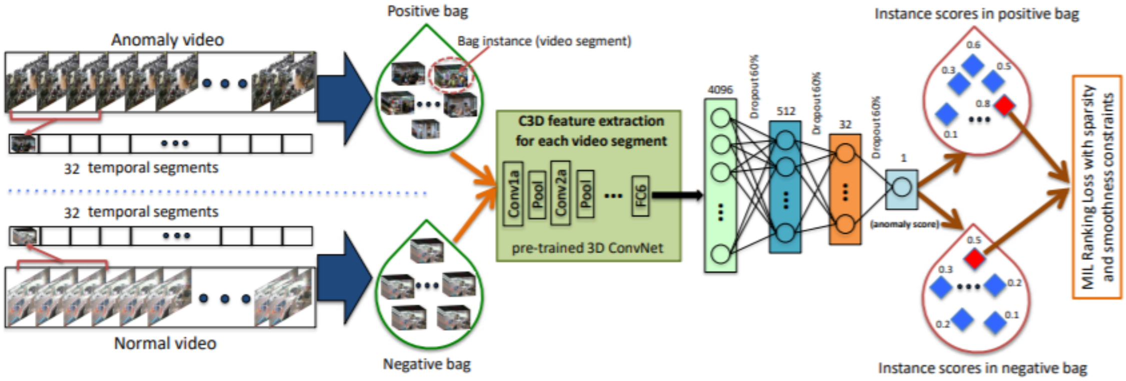
- **Hardware Requirements**
  - Personal Computer with >=8GB RAM
  - Dedicated GPU for better performance

METHODOLOGY USED

Our solution is AI based, and involves computer vision (CV) and deep learning (DL) for monitoring and detecting any suspicious activity. To implement this, our methodology is:

In this project, we first learn anomalies by exploiting both normal and anomalous videos. To avoid annotating the anomalous segments or clips in training videos, which is very time consuming, we propose to learn anomaly through the deep multiple instance ranking framework by leveraging weakly labeled training videos, i.e., the training labels (anomalous or normal) are at video-level instead of clip-level. In our approach, we consider normal and anomalous videos as bags and video segments as instances in multiple instance learning (MIL), and automatically learn a deep anomaly ranking model that predicts high anomaly scores for anomalous video segments. We use this model for general anomaly detection considering all anomalies in one group and all normal activities in another group. Furthermore, we introduce sparsity and temporal smoothness constraints in the ranking loss function to better localize anomaly during training.

Flow of the proposed model:



The flow diagram of the proposed anomaly detection approach is shown above. Given the positive (containing anomaly somewhere) and negative (containing no anomaly) videos, we divide each of them into multiple temporal video segments. Then, each video is represented as a bag and each temporal segment represents an instance in the bag. After extracting C3D features for video segments, we train a fully connected neural network by utilizing a novel ranking loss function which computes the ranking loss between the highest scored instances in the positive and the negative bag.

IMPLEMENTATION DETAILS

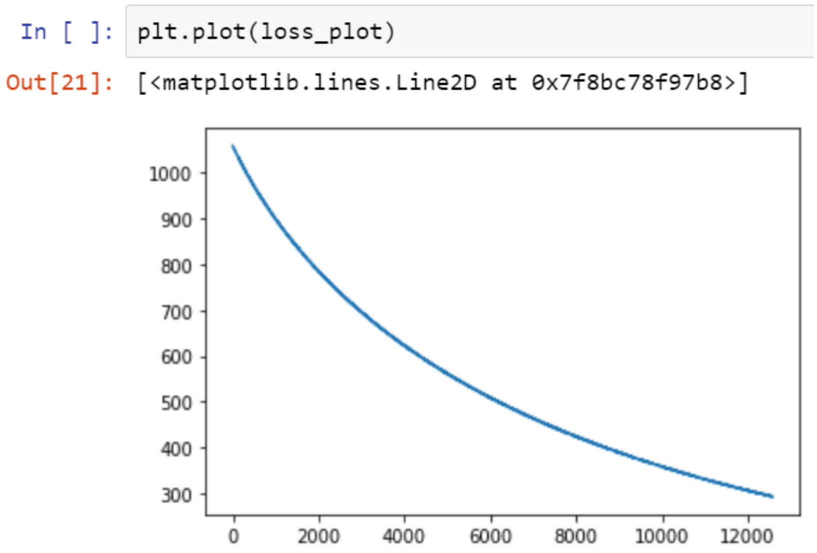
1. We extract visual features from the fully connected (FC) layer FC6 of the C3D network. Before computing features, we re-size each video frame to 240 × 320 pixels and fix the frame rate to 30 fps. We compute C3D features for every 16-frame video clip followed by l2 normalization. To obtain features for a video segment, we take the average of all 16-frame clip features within that segment.
2. The first FC layer has 512 units followed by 32 units and 1 unit FC layers. 60% dropout regularization is used between FC layers. We use ReLU activation and Sigmoid activation for the first and the last FC layers respectively, and employ Adagrad optimizer with the initial learning rate of 0.001.
3. The parameters of sparsity and smoothness constraints in the MIL ranking loss are set to $\lambda 1=\lambda 2 = 8\times10-5$ and $\lambda 3 = 0.01$ for the best performance.
4. We divide each video into 32 non-overlapping segments and consider each video segment as an instance of the bag. The number of segments (32) is empirically set.
5. We randomly select 30 positive and 30 negative bags as a mini-batch. We compute gradients by reverse mode automatic differentiation on computation graph using Theano.
6. Then we compute loss, and back-propagate the loss for the whole batch for minimizing it.
7. We use frame based receiver operating characteristic (ROC) curve and corresponding area under the curve (AUC) to evaluate the performance of our method.

RESULTS

Screenshots:

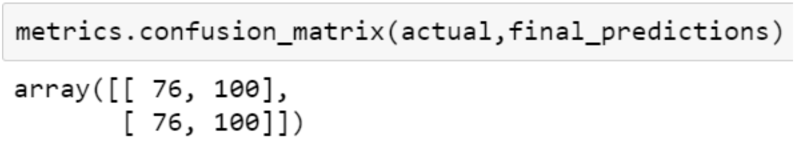
- Back-end:

1. Training the model, corresponding values of the loss function:



```
In [ ]: plt.plot(loss_plot)
Out[21]: [<matplotlib.lines.Line2D at 0x7f8bc78f97b8>]
```

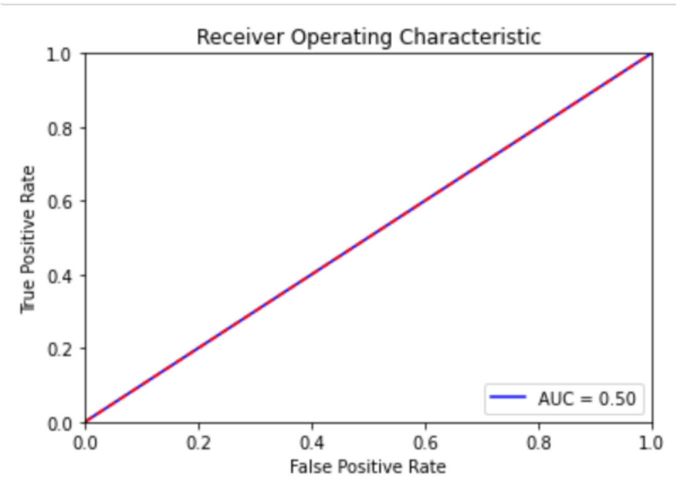
2. Testing the model, confusion matrix:



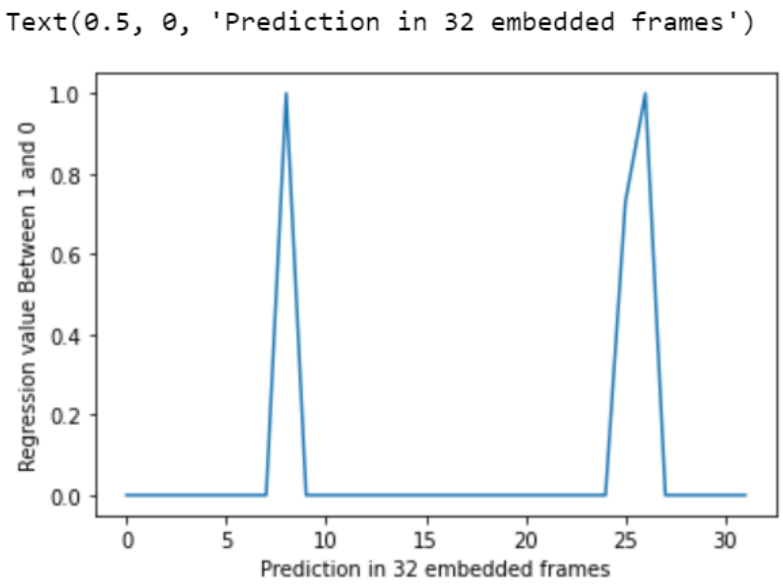
```
metrics.confusion_matrix(actual,final_predictions)

array([[ 76, 100],
       [ 76, 100]])
```

3. AUC:

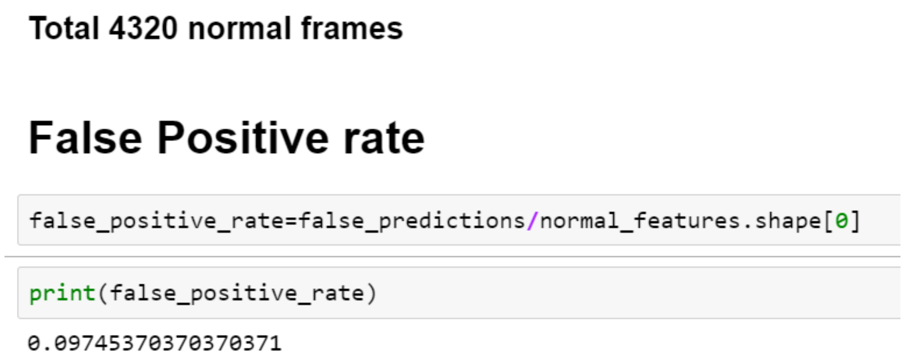


4. Graphically depicting the classification of frames:



```
Text(0.5, 0, 'Prediction in 32 embedded frames')
```

5. False Positive Rate:



Total 4320 normal frames

False Positive rate

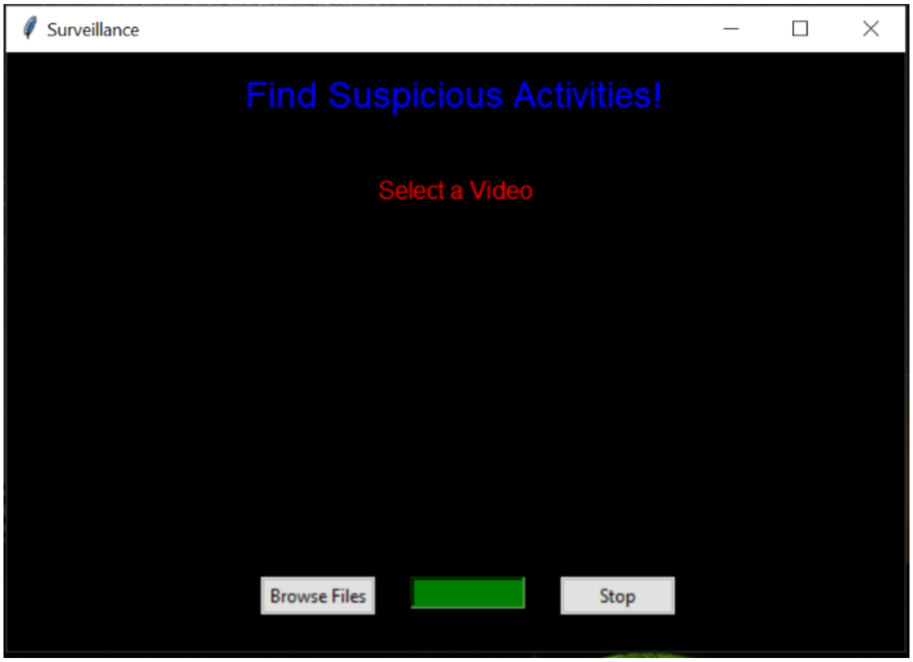
```
false_positive_rate=false_predictions/normal_features.shape[0]
```

```
print(false_positive_rate)
```

```
0.09745370370370371
```

- Front-end:

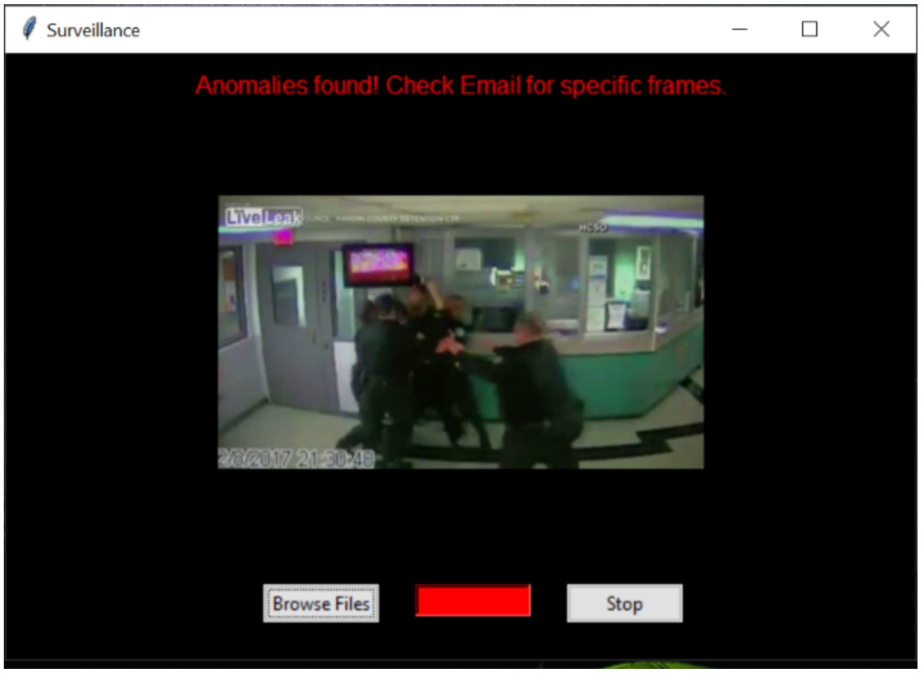
1. Initial Page:



Find Suspicious Activities!

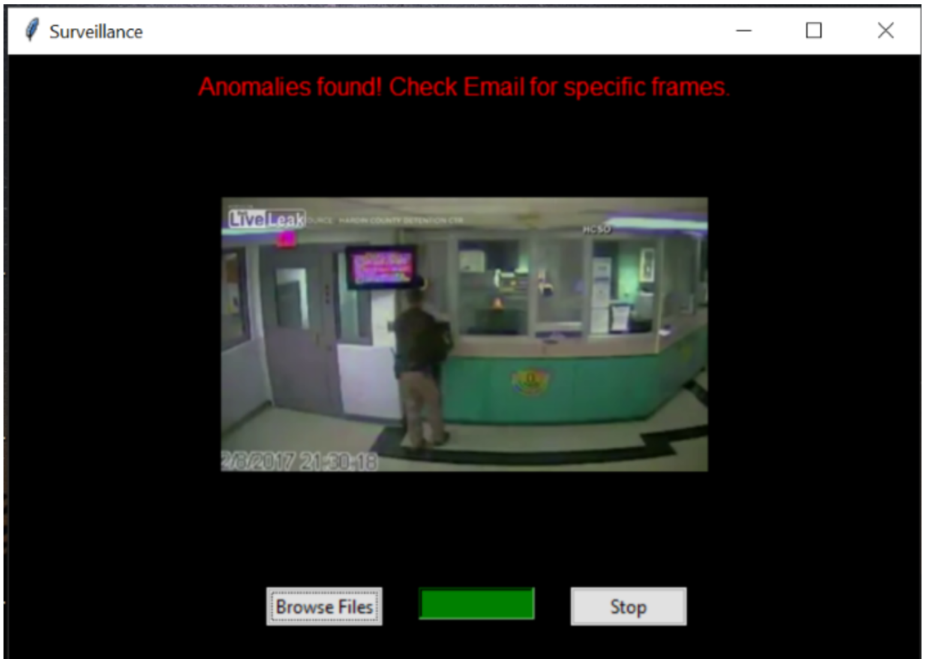
Select a Video

2. After uploading the video, red bar indicates some unusual activity (fight) is occurring right now, corresponding frames are sent in the mail:



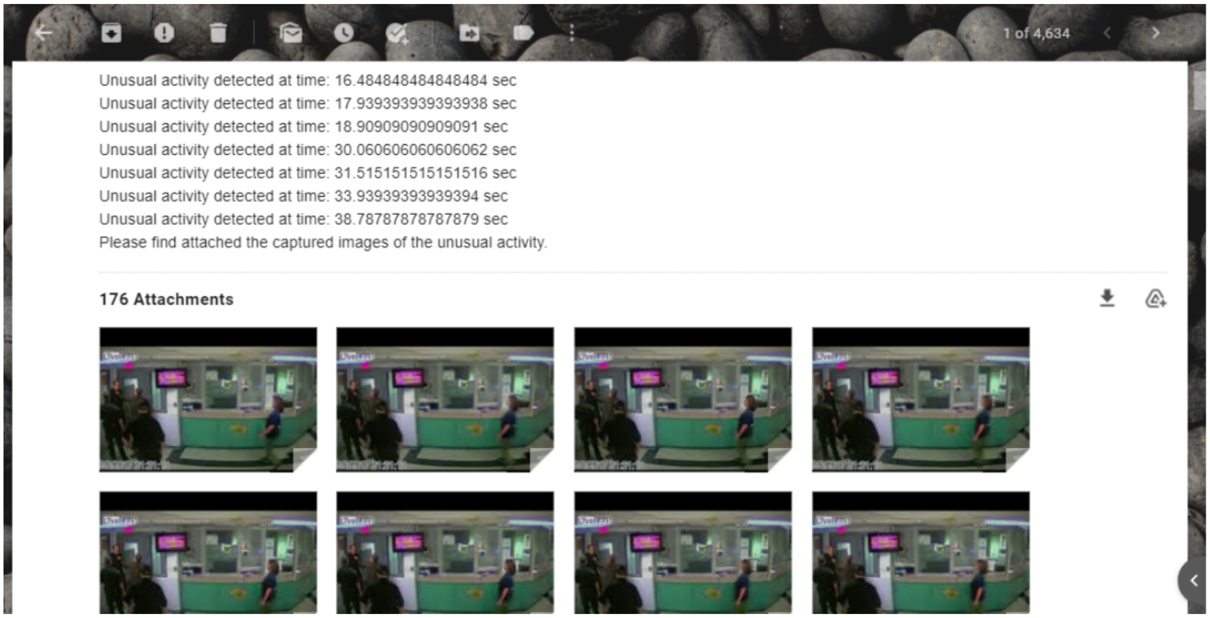
Anomalies found! Check Email for specific frames.

3. After uploading the video, green bar indicates normal activity is going on right now:



Anomalies found! Check Email for specific frames.

- Reporting:

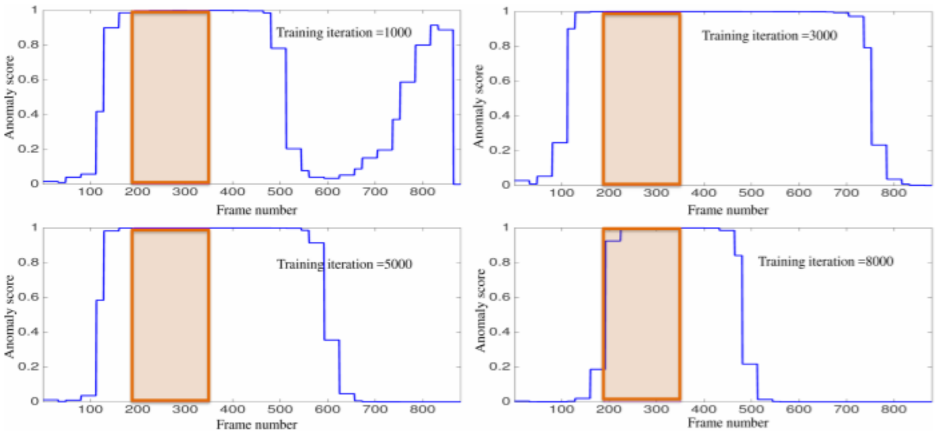


```
Unusual activity detected at time: 16.484848484848484 sec
Unusual activity detected at time: 17.939393939393938 sec
Unusual activity detected at time: 18.90909090909091 sec
Unusual activity detected at time: 30.060606060606062 sec
Unusual activity detected at time: 31.515151515151516 sec
Unusual activity detected at time: 33.93939393939394 sec
Unusual activity detected at time: 38.78787878787879 sec
Please find attached the captured images of the unusual activity.
```

176 Attachments

E-mail sent to the admin with snapshots where anomalies were detected

Analysis: The underlying assumption of the proposed approach is that given a lot of positive and negative videos with video-level labels, the network can automatically learn to predict the location of the anomaly in the video. To achieve this goal, the network should learn to produce high scores for anomalous video segments during training iterations.



Evolution of score on a training video over iterations. Colored window represents ground truth (anomalous region). As iteration increases, our method generates high anomaly scores on anomalous video segments and low scores on normal segments.

WORK DIVISION

- Adit Goyal: Research and analysis for the methodology and the corresponding algorithm, and frontend-backend integration.
- Amritanshu Agrawal: Research for methodology, and entire front-end development.
- Shreyansh Das: Setting up environment, and implementing the algorithm considered for back-end.

CONCLUSION

We propose a deep learning approach to detect real-world anomalies in surveillance videos. Due to the complexity of these realistic anomalies, using only normal data alone may not be optimal for anomaly detection. We attempt to exploit both normal and anomalous videos. To avoid labor-intensive temporal annotations of anomalous segments in training videos, we learn a general model of anomaly detection using deep MIL framework with weakly labeled data.

This approach can be used for classification of anomalies in several categories based on the dataset used to train the model. Furthermore, with the relevant dataset, reporting of the such incidents can also be streamlined and detailed using NLP to make this project more user-friendly. Hence, the major limitation faced by us was data constraints, and lack of time and power for the modeling of huge datasets.

References:

1. https://openaccess.thecvf.com/content_cvpr_2018/papers/Sultani_Real-World_Anomaly_Detection_CVPR_2018_paper.pdf

2. https://www.github.com/facebook/C3D

Model and Data Source:

1.https://www.dropbox.com/sh/75v5ehq4cdg5g5g/AABvnJSwZI7zXb8_myBA0CLHa?dl=0