Crime Analysis with Computer Vision

Kim Jae Seong, Gao Zichen and Khen Cohen
Department of Electrical and Electronics Engineering, Tel Aviv University

Abstract—With an increase in number of surveillance cameras installed in the public, it has become a more difficult mission for the authorities to constantly monitor the screen. To solve this issue, we present a "Crime Analysis Model" which is able to classify 10 different types of crime efficiently using the convolutional neural network (CNN) given any surveillance video footage as its input. We have utilized two separate deep learning models within the Crime Analysis model, namely (1) anomaly detector (2) crime classifier (which is trained based on the human-action recognition). The Crime Analysis model takes any form of surveillance video footage as its input. The input first gets passed to the Anomaly Detector which helps to distinguish time frames at which anomalies occur. The corresponding time frames at which the anomalies occur gets passed to the crime classifier. The change of position of the skeletons within the video are analyzed and the model outputs one of the following crime categories: (1) Abuse (2) Arson (3) Assault (4) Burglary (5) Fighting (6) Robbery (7) Shooting (8) Shoplifting (9) Stealing (10) Vandalism

Index Terms—Convolutional Neural Network, Anomaly Detection, Human-action Recognition, Crime Classification

1 Introduction

1.1 Background

With an increasing number of surveillance cameras installed in public and private places, it has become a more challenging task for the authorities to constantly monitor all the cameras. With such a vast amount of data, it would be a better choice to let the computers do the monitoring process. The purpose of the project is to create a deep learning model which could self-monitor all the surveillance videos and alert the user of any anomalies. Furthermore, the model has the capability to further classify the anomaly detected into either one of 10 different crime categories.

There are two functions within the Crime Analysis model: (1) Anomaly Detection and (2) Crime Classifier. Each functionality is a separate model and is a huge research topic by itself. For the Anomaly Detection, there have been many approaches to increasing the AUC (Area Under the Curve) Score, which tells about the accuracy in detecting anomalies. For the Crime Classification, many projects were developed to predict potential crimes that might take place.

Nonetheless, there have been close to no attempts to combine the two models to detect crimes better and faster. We have taken a noble approach in combining the two models, hoping the combination of the two would bring a more efficient result in crime detection and classification.

1.2 Anomaly Detector

Anomaly Detector is a CNN used for detecting anomalies that are present in the input videos. Given an video input, it produces an anomaly score at each time in millisecond. The anomaly score varies between 0 to 1, 0 being an absolute normal case and 1 being an absolute abnormal case.

The main role of the anomaly detector is to preprocess the video for the next model, which is the crime classifier. We do not want the crime classifier to go over the video of entire length and constantly predict the crime that is taking place at the scene. Instead, we would like to only pass the time frames at which the anomaly detector thinks that has a high chance of crime taking place. This would reduce the amount of processing for the crime classifier.

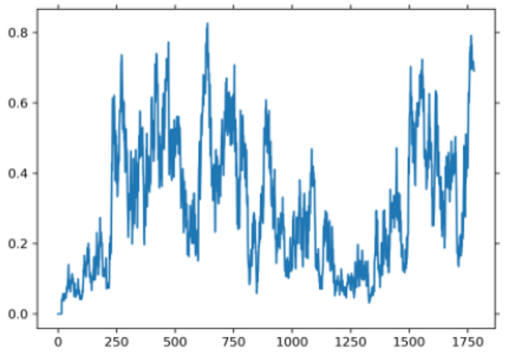

Fig. 1: Example of anomaly score generated by the anomaly detector [1]. The X-axis is the time in milliseconds. The Y-axis is the anomaly score ranging between 0 to 1.

For our crime analysis model, we have decided to use pre-existing anomaly detector. We used the model from "Real-world Anomaly Detection in Surveillance Videos" [2], which was presented in Computer Vision and Pattern Recognition Conference (CVPR) from 2018 [3].

1.3 Crime Classifier

We now have the information regarding the time frames at which the anomalies occur. It is our time to further analyze the video at those time frames. To classify the type of crime, it is essential to analyze the movements of human bodies. The best way would be to take a look at the skeleton diagram of the people in the video and see if any certain movements are associated with a certain type of crime. This is where the Crime Classifier comes into play.

The Crime Classifier model is built based on the human-action recognition model. The human-action recognition model is capable of identifying the human body and applying the skeleton

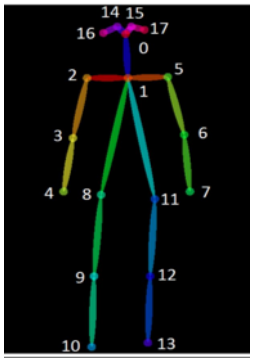

Fig. 2: Skeleton diagram [4] - it contains 18 nodes (0-17). Each node has a X-Y coordinate information

diagram. The X-Y positions of each node are extractable. Essentially, the crime classifier model seeks for the change in those X-Y position coordinates to check the trajectory of certain movements and come up with the closest classification category.

1.4 UCF-Crime Dataset

The UCF-Crime dataset [5] is a collection of real-life surveillance videos. The dataset contains 128 hours of videos and consists of 1900 long and untrimmed videos. There are 13 categories of crime within the dataset: (1) Abuse (2) Arrest (3) Arson (4) Assault (5) Road (6) Accident (7) Burglary (8) Explosion (9) Fighting (10) Robbery (11) Shooting (12) Stealing (13) Shoplifting

There are lots of other datasets that are associated with crime but many of them are acted in a fixed environment by the actors. It may seem that the set is perfectly constructed and the actions resemble real life; however, the lighting and the angles may be too perfect compared to the real-life video footage. For instance, if the crime is happening at night, there may not be enough lighting to perfectly capture the movements. Another example would be the existence of surrounding obstacles. The acting is set at a perfect angle so that the skeleton would be easily constructed; However, in reality, there may be obstacles that block the view of the human body and this will bring difficulty in constructing the skeleton diagrams. In short, those datasets minimize the real-life aspect thus the deep learning model trained with them may not be the most optimum for detecting and classifying real-life crime surveillance inputs.

Both the anomaly detector and the crime classifier were trained on the UCF-Crime dataset. Thus, we have made the effort to let the model take into account real-life aspects – lighting, angle, obstacles, etc. - as much as possible.

You might have noticed that we have mentioned previously that our model produces a classification of 10 different types of crime but the dataset contains 13 different types. This is because we have removed "(5) Road (6) Accident (8) Explosion". There aren't many scenes of the human body in those categories. Road, for example, contains more vehicles in the scene than the human body. For such reason, we have removed those 3 categories.

2 Related Work

2.1 Anomaly Detection Related Work

Anomaly detection has been a field of research by itself. Many techniques have been built throughout the years to improve the accuracy in detecting anomalies. Here is the list of some of the techniques: (1) C3D video feature extraction to identify the characteristics in the video (2) weakly supervised learning where instead of labelling frame by frame on where the anomaly is, the model is trained with video that are known to contain anomaly but not given specific location of the occurrence (3) Generative Neural Networks (GNN) [6] to create new images and distinguish between fake and real data. All the functionalities above have been implemented as a part of the process for the anomaly detection model.

AUC ↑	Decidability	EER	Paper	Code	Result	Year
0.906	1.386	0.160	Weakly and Partially Supervised Learning Frameworks for Anomaly Detection			2020
0.892	0.804	0.186	Real-world Anomaly Detection in Surveillance Videos			2018
0.610	0.323	0.427	Generative Neural Networks for Anomaly Detection in Crowded Scenes			2018
0.541	0.059	0.480	Abnormal Event Detection in Videos using Spatiotemporal Autoencoder			2017
0.533	0.147	0.484	Abnormal Event Detection in Videos using Generative Adversarial Nets			2017
0.528	0.194	0.466	Learning Temporal Regularity in Video Sequences			2016

Fig. 3: List of anomaly detection models listed with highest accuracy to the lowest. [7]

The current state of the art in anomaly detection is "Weakly and Partially Supervised Learning Frameworks for Anomaly Detection" by Bruno Manuel Degardin [8]. It has the best measure of accuracies. High accuracy means high AUC (Area Under the Curve), high decidability and low EER (equal error rate). This model utilizes all the techniques previously developed for anomaly detection.

Our entire Crime Analysis Model was built with PyTorch, while the above anomaly detector was built with TensorFlow. Thus, we decided to use the second best model which is "Real-world Anomaly Detection in Surveillance Videos".

Many similar projects have been built to classify different types of crime. Many projects utilized spatio-temporal data approach [9] in solving the classification problem. The dataset used were CSV files which contained spatio-temporal information. Many classifiers also used different dataset other than UCF-Crime dataset. Some have utilized UBI-Fights [10] and there are many other regional datasets – Crime in Vancouver [11], Crime in England and Wales [12], etc. In our model, we took CNN approach in solving the classification problem, focusing heavily on tracking the trajectory of the movements through human-action recognition model.

3 Method

3.1 Anomaly Detector Implementation

Multiple Instance Learning (MIL) [13] approach is used to train the Anomaly Detection Model. MIL is a technique used in machine learning where we look at segments of the video first and then make conclusion based on the results of the segments. For better understanding, here are some important definitions.

For MIL, each data is split into multiple segments. Each segment of the video is defined as an instance. The entire video is made of the collection of all the instances and this is what we call a bag. Therefore, simply put, instance is a video segment and bag is a collection of the instances, which make an entire video.

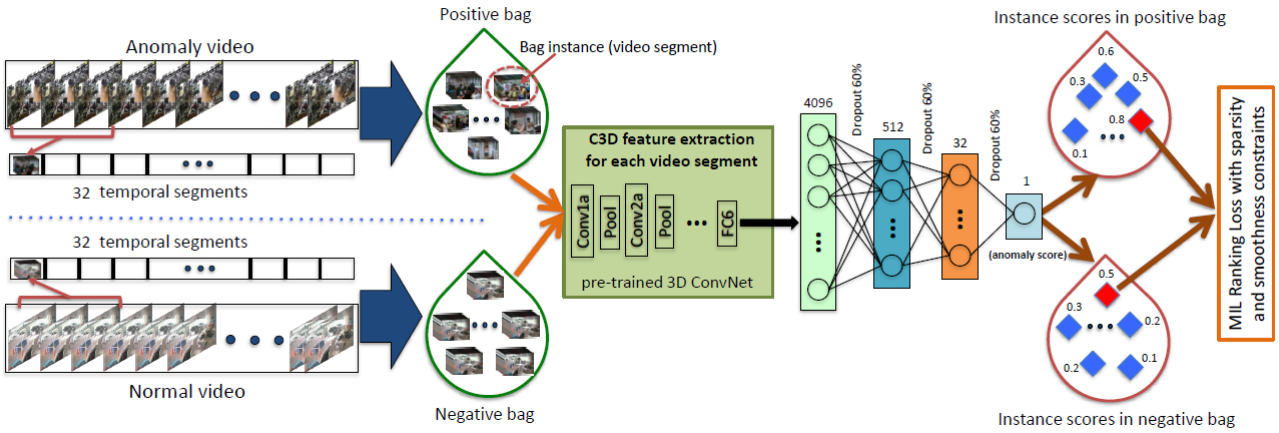

Fig. 4: Anomaly Detector implementation method [2]. Both the anomalous (positive) and normal (negative) videos are segmented. The segments for a bag and passed to the C3D feature extractor. The outputs from the extractor are passed to several Fully Connected layers and anomaly scores are labeled for each segment. Compare the highest score from each bag and compute the difference for the MIL ranking loss error. Repeat the process to minimize the error.

Now, let's discuss why looking into segments of the video is important. In the case of anomaly detection, let's say you are given an input video of robbery. Usually the surveillance video includes the scene before the crime and after the crime. We do not know exactly where the robbery is taking place. Therefore, to understand specific time frame at which an anomaly seems to be taking place, it is important to split the video into segments and look at them one by one and then make a conclusion as a whole.

Now that we know the terminology, here is the process of how the anomaly detection model is trained. There are two types of videos that are fed into the model – the video with anomaly (positive) and without, which is normal (negative). If there is anomaly in the video, we call it positive. If the video scenes are normal, we call it negative. The segments of each type of video is put into a positive bag and a negative bag respectively. The bag goes through C3D feature extraction process for each segment. The result of the feature extractor goes through multiple Fully Connected (FC) layers and anomaly scores are produced for each segment. Now we have two bags (positive and negative) with anomaly scores labeled for each segment. We compare the segment with highest anomaly score from each bag and computer the error. The is back-propagated within the model and the weights are updated to reduce the error as much as possible.

After model is implemented, we input a surveillance video. The model generates anomaly score plot with anomaly score vs time in millisecond. We manually set a threshold telling that once the model crosses this line, we will consider it as an anomaly. We look at the time stamps where anomaly score goes beyond the threshold. We record those time stamps and send the information to the crime classifier.

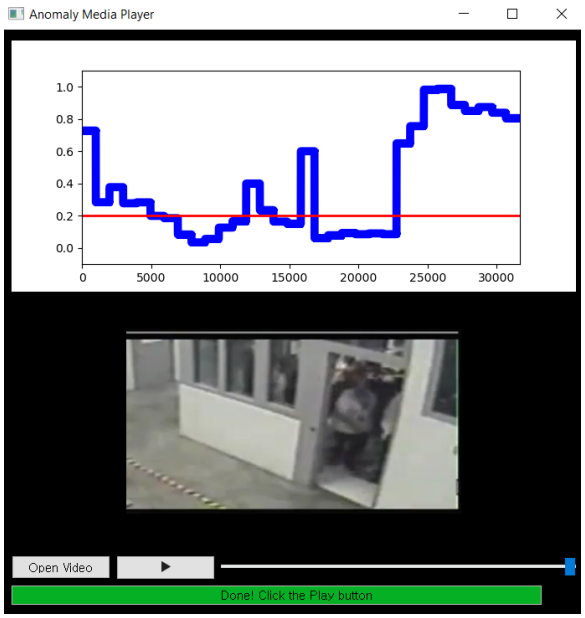

Fig. 5: Ex: Anomaly Score vs time in ms. The blue line represents the anomaly score and the red line represents the threshold. In our implementation, we used threshold=0.2. Time stamps at which the anomaly score crosses the threshold will be recorded and passed on to the crime classifier for classification.

3.2 Crime Classifier Implementation

The crime classifier is built based on the human-action recognition model [14]. We have utilized two functionalities from this model – pose estimation and people tracking. The pose estimation is built with a library called Trtpose [15], which heavily depends on TensorRT by NVDIA. TensorRT provides accelerated computation which increases accuracy of pose estimation and makes the prediction faster. People tracking is made possible with a library called "Yolov5+StrongSORT with OSNET" [16]. It utilizes YOLOv5 for object detection and StrongSORT for tracking the objects.

Pose estimation and people tracking are insufficient to build a classifier. It provides us the capability to build the skeleton model and track the nodes. With this information, we will examine the trajectory of the nodes and see if we can match the characteristic to those of 10 different crime categories.

First we need to understand the information of the input that

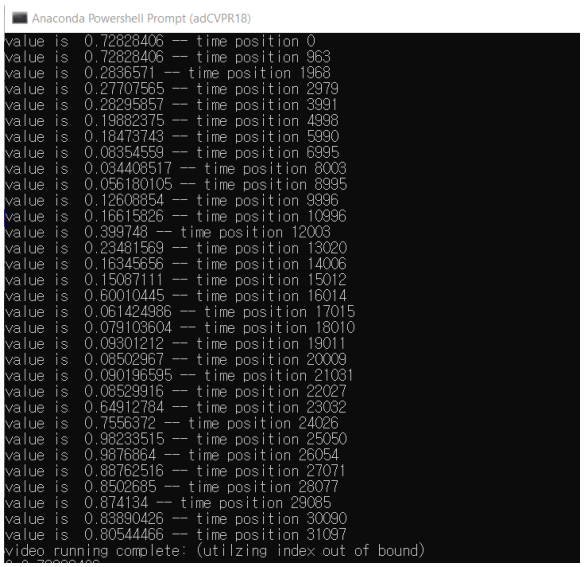

Fig. 6: Ex: anomaly score with corresponding time in ms

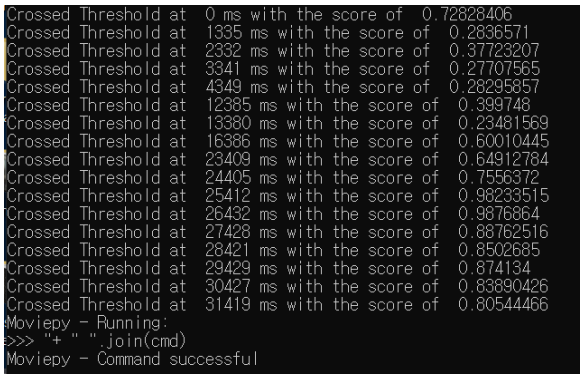

Fig. 7: Ex: anomaly score with corresponding time in ms that are above the threshold

going into the classifier.

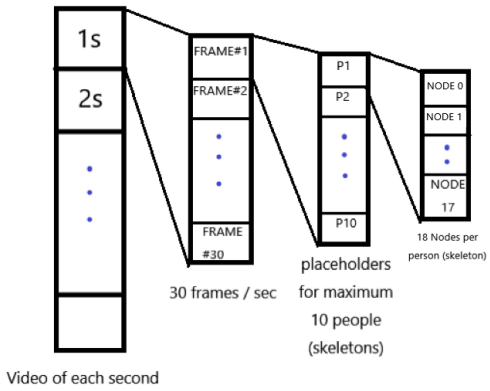

Fig. 8: input dimension

The input videos are 30 frames/sec and for each frame and we want to analyze the skeleton nodes inside. We have set placeholders for a maximum of 10 people. In other words, a maximum of 10 skeletons could be detected simultaneously and their X-Y positions could be traced. As previously mentioned, each skeleton has 18 nodes and the detection of the skeleton is done using the human-action recognition model. In the next paragraph, we will explain about the sliding window method which is an algorithm for training our classifier model. However, for the sake of explanation of the input vector to the classifier, the size of the window is 60 frames. Therefore, the input vector will have a dimension of 60 frame x 10 people x 18 nodes x 2 X-Y position values.

Now we have understanding of the input, let's see how the algorithm works to process the input.

We came up with a "sliding window" method for the processing of the input data. The window is divided into 60 slots and each slot could contain the skeleton information within each frame. We process the video by sliding the video frame by frame. Something important to note is that first frame of the video will be allocated to the last slot in the window and will be filled consecutively by sliding to the right. Once the window is full, we use First in First Out (FIFO) approach to continue processing the data.

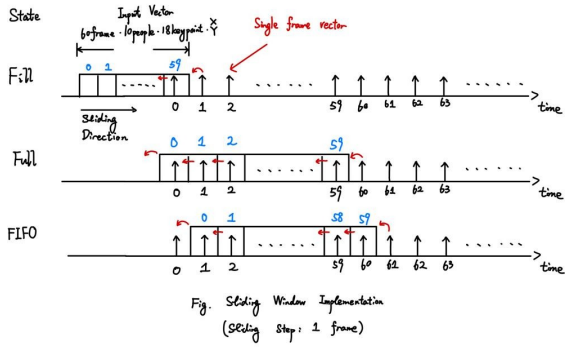

Fig. 9: Classifier Implementation Flow

Here are the details for the layers used in training the classifier. The classifier consists of 4 Fully Connected (FC) layers. 1st layer: 21600 $\rightarrow$ 1020, 2nd layer: 1020 $\rightarrow$ 510, 3rd layer: 510 $\rightarrow$ 50, and 4th layer: 50 $\rightarrow$ 10. The activation function is Rectified Linear Unit (ReLu). The loss algorithm used is Cross Entropy Loss. The optimization algorithm used is Stochastic Gradient Descent (SGD).

As you can see the dimension of the last layer of FC is 10. This is the 10 categories of crime we are classifying. When we input a crime scene to the classifier, it produces a percentage distribution of how this input video is likely to fall under each category. We choose the category with highest percentage and display that as the final result.

For the training, we used the UCF-Crime dataset and utilized 80% of the dataset as training set and 20% as the test set.

Fig. 10: Output of the crime labeller. A scene acted for robbery and classification of robbery as its output

4 Results

4.1 Accuracy of the Anomaly Detector

Anomaly Detector detects whether an anomaly has occurred or not. Although Anomaly Detector outputs anomaly score between 0 to 1 at different time segments, we will set a threshold to determine the standard of anomaly. If the score passes the threshold, it is considered as an anomaly and the relevant data will be passed on to the classifier. For our project, we have determined the threshold to be 0.2.

Therefore, essentially we could think of the Anomaly detection as a binary classification model as we are trying to distinguish between anomalous and normal time frames. To measure the accuracy of such model, we need to look at Receiver Operating Characteristics (ROC) and Area under the curve(AUC) [17].

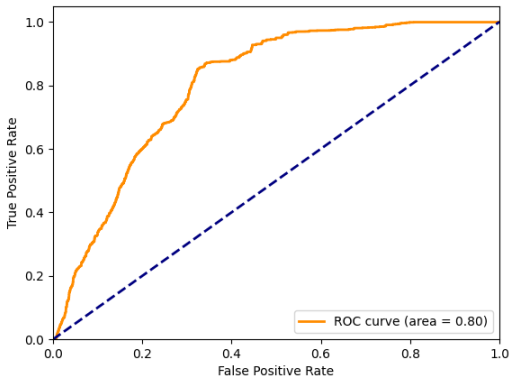

Fig. 11: tpr vs fpr. The orange cureve is the ROC. AUC is the area under the ROC. The higher the AUC, the better it is for the anomaly detector to correctly produce anomaly score at different time frames. In this plot, AUC is 80%.

```
Epoch: 97
('loss = {}', 0.42102776522989627)
('auc = ', 0.8320011379921686)

Epoch: 98
('loss = {}', 0.41414477041474096)
('auc = ', 0.817898709048825)

Epoch: 99
('loss = {}', 0.42418995941126786)
('auc = ', 0.8212386611475362)

Epoch: 100
('loss = {}', 0.41555263929896885)
('auc = ', 0.8154726737216201)
```

Fig. 12: AUC of the Anomaly Detector is around 80%.

4.2 Accuracy of the Crime Classifier

The accuracies are different for each category and it varies from 31.7% (Shooting) to 87.5% (Arson). The reason for the variation in accuracy for each category could be related to the number of samples we have for each category, as having greater number of samples is more likely to bring higher accuracy.

However, there seem something strange in the number of samples. How could Arson have 87.5% accuracy while it has one of the fewest number of samples? This is because we have utilized data augmentation to account for the lack of samples for some of the categories.

Data augmentation was implemented for the first 5 categories: (1) Abuse (2) Arson (3) Assault (4) Shooting (5) Vandalism. To apply up-sampling for the data, we have made small random shifts to the skeleton nodes. The movements do not deviate much from the original trajectory of the skeleton nodes but could aid in bringing in more data. This is also what makes our classifier unique.

5 Discussion

5.1 Imperfection

There exist imperfections in the crime analysis model. Sometimes the anomaly detector is not able to detect any anomalies given an anomalous video and crime classifier gives wrong classification for the labelled input. Here are some of the examples of imperfections and explanation for the causes.

5.1.1 Imperfection in Anomaly Detector

Left is an adult stepping on the child. Right is a scene where a person is hitting the dog with a stick. Both are videos in the category of Abuse. Even though the act of crime is evident from the video, the anomaly score falls below 0.2 and stays very close to 0.0, which means perfect anomaly. Obviously, this is an inaccurate score.

5.1.2 Imperfection in Crime Classifier

As shown in the previous result section, there are inaccuracies in the crime classifier. For instance, we have acted a scene for robbery, however, we have gotten burglary as the result. This could show the challenging aspect of the crime classification. Because many of the crimes share similarities, there are no absolute action or movement trajectory to define a class. For example, fighting, abuse, and assault could all involve punching and kicking. It is difficult for the classifier to be very accurate in those categories.

Another problem that arises in classification is overlap of multiple criminal actions. Let's say the main crime happening at the scene is arson. Imagine people fighting during an argument and suddenly a person decides to commit arson. Overall, the main crime would be arson but the classifier might focus on the fighting scene and output fighting as the final outcome.

5.2 Improvement

Anomaly Detector – The Anomaly Detector uses C3D feature extractor for extracting visual data from the video. There are a few alternatives to C3D feature extractor. I3D [18] and MFNET [19] feature extractor are also gaining popularity. Trying the alternatives may bring up the accuracy in detecting anomalies.

Crime Classifier – (1) Train the model with higher number of data samples. With increase in the number of data samples, the model has more data to analyze; therefore, increasing the chance of better classification. In our classifier, we saw that there were uneven number of samples for each category. Although we have used data augmentation for up-sampling [20] purposes, it would be much better to have greater and even number of samples for each category.

(2) Improve the detection of the trajectory of the movements is by using relative position of the skeleton nodes rather than tracking the absolute. We have been tracking how each of the nodes been moving around individually, however, it might give better results when we see how node positions change relative to a reference point.

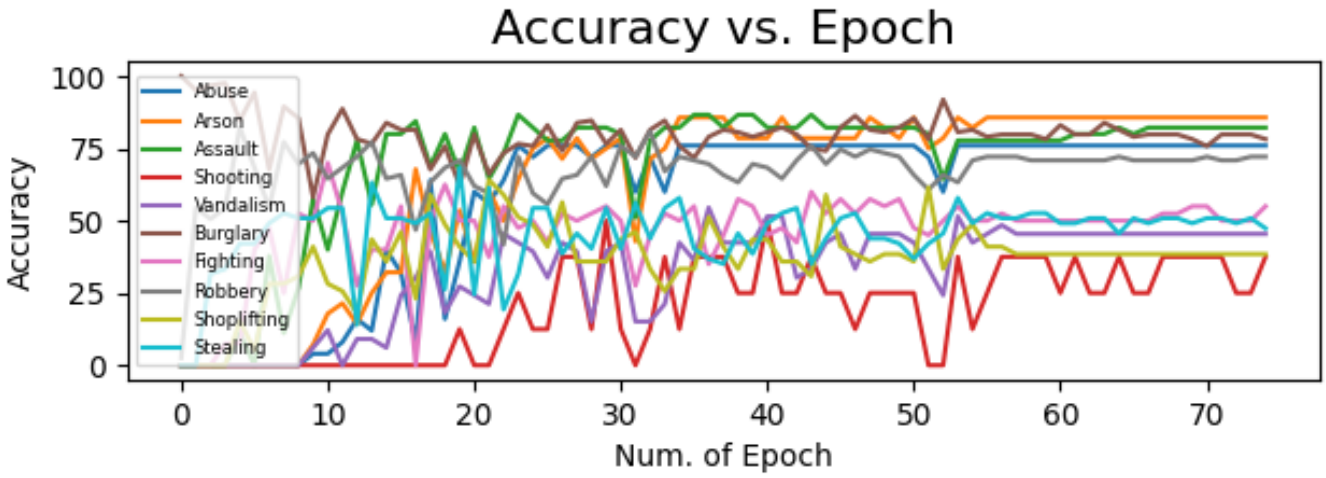

Fig. 13: Classification accuracy for each category

Arson	Assault	Burglary	Stealing	Robbery
87.5%	82.3%	78.6%	76.1%	73.4%
Fighting	Stealing	Vandalism	Shoplifting	Shooting
57.3%	50.6%	45.2%	38.4%	31.7%

Fig. 14: Crime Classifier accuracy table for each category

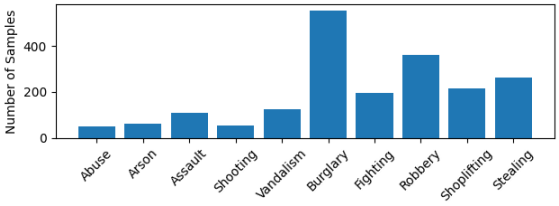

Fig. 15: Number of Samples used to training the Crime Classifier

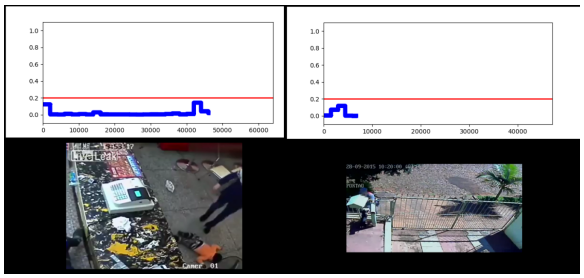

Fig. 16: Anomaly score of Abuse017 and Abuse028 respectively

Fig. 17: Failed case of the crime labeller. A scene acted for robbery but getting other classes - fighting, shooting, shoplifting, as its result

(3) Apply multi-labelling instead of single label. The crime analysis model was designed to list the probability for each category and choose the category with the highest probability given an input video. As mentioned in the imperfections section, there could be a lot of overlap of different crimes present in the video. Instead of giving a single absolute label, we can give multiple labels to a given video to better describe all the actions that are taking place. For example, we can say "video contains fighting, assault and arson" for better analyzing the crime scene.

(4) Use larger network for skeleton model to improve the accuracy and stability. Now we are using trtpose of size 256. It's backbone densenet121 [21] of 256x256 epoch 160. We could increase the size.

6 Conclusion

The purpose of the Crime Analysis model is to identify anomalies in the surveillance videos and provide a classification to the crime taking place during the anomalous time periods. The output of the crime analysis model is one of the following crime types: (1) Abuse (2) Arson (3) Assault (4) Burglary (5) Fighting (6) Robbery (7) Shooting (8) Shoplifting (9) Stealing (10) Vandalism

The UCF-Crime dataset was used to resemble the real life crime scenes as much as possible. We have utilized the pre-exisiting anomaly detector to identify the abnormal and normal time frames and primarily focus on the abnormal time frames. The classifier utilizes human-action recognition model to identify the skeleton information of the people involved in the crime and analyzes the crime type based on the trajectory of the skeleton nodes. Sliding window of size 60 frames was used to analyze the skeleton movement. Due to unevenness in the sample data, we have utilized data augmentation for up-sampling.

7 References

References

[1] seominseok0429, "Seominseok0429/real-world-anomaly-detection-in-surveillance-videos-pytorch: Real-world anomaly detection in surveillance videos- pytorch re-implementation." [Online]. Available: https://github.com/seominseok0429/Real-world-Anomaly-Detection-in-Surveillance-Videos-pytorch 1

[2] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 1, 3

[3] E. Kosman, "Pytorch implementation of real-world anomaly detection in surveillance videos," https://github.com/ekosman/AnomalyDetectionCVPR2018-Pytorch, accessed: 2022-05-15. 1

[4] B. Raj and Y. Osin, "An overview of human pose estimation with deep learning." [Online]. Available: https://www.kdnuggets.com/2019/06/human-pose-estimation-deep-learning.html 2

[5] W. Sultani, C. Chen, and M. Shah. [Online]. Available: https://www.crcv.ucf.edu/research/real-world-anomaly-detection-in-surveillance-videos/ 2

[6] T. Wang, M. Qiao, Z. Lin, C. Li, H. Snoussi, Z. Liu, and C. Choi, "Generative neural networks for anomaly detection in crowded scenes," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1390–1399, 2019. 2

[7] "Papers with code - ubi-fights benchmark (abnormal event detection in video)." [Online]. Available: https://paperswithcode.com/sota/abnormal-event-detection-in-video-on-ubi 2

[8] B. M. Degardin, "Human activity analysis: Iterative weak/self-supervised learning ... - ubi," 2020. [Online]. Available: https://www.di.ubi.pt/~hugomcp/doc/ijcb2020_degardin.pdf 2

[9] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. 2

[10] B. Degardin and H. Proença, "Human activity analysis: Iterative weak/self-supervised learning frameworks for detecting abnormal events," in *2020 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, pp. 1–7. 2

[11] "Crime statistics," May 2022. [Online]. Available: https://vpd.ca/crime-statistics/ 2

[12] M. Elkin, "Crime in england and wales," Apr 2022. [Online]. Available: https://www.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/datasets/crimeinenglandandwalesappendixtables 2

[13] M.-A. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon, "Multiple instance learning: A survey of problem characteristics and applications," *Pattern Recognition*, vol. 77, p. 329–353, 2018. 2

[14] Z. Moe, "Cv-zmh/human-action-recognition: Multi person skeleton based action recognition and tracking." [Online]. Available: https://github.com/CV-ZMH/human-action-recognition 3

[15] "Creating a human pose estimation application with nvidia deepstream," Oct 2021. [Online]. Available: https://developer.nvidia.com/blog/creating-a-human-pose-estimation-application-with-deepstream-sdk/ 3

[16] M. Brostrom, "Mikel-brostrom/yolov5-strongsort-osnet: Real-time multi-camera multi-object tracker using yolov5 and strongsort with osnet." [Online]. Available: https://github.com/mikel-brostrom/Yolov5-StrongSORT-OSNet 3

[17] J. Huang and C. Ling, "Using auc and accuracy in evaluating learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, p. 299–310, 2005. 5

[18] X. Wang, Z. Miao, R. Zhang, and S. Hao, "I3d-lstm: A new model for human action recognition," *IOP Conference Series: Materials Science and Engineering*, vol. 569, no. 3, p. 032035, 2019. 5

[19] S. Gong, E.-B. Bourennane, and X. Yang, "Mfnet: Multi-feature convolutional neural network for high-density crowd counting," in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2020, pp. 0384–0390. 5

[20] V. Kumar, H. Glaude, C. de Lichy, and W. Campbell, "A closer look at feature space data augmentation for few-shot intent classification," *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, 2019. 5

[21] W. Huang, G. Peng, and X. Tang, "A limit of densely connected convolutional networks v1," 2019. 6