# Activity Recognition: Honours Project Backlog

Thomas Scholtz, Stellenbosch University

February 2021

## Contents



## 1 Introduction

The following document outlines a project backlog for the implementation and proposed improvement of an anomaly detection framework devised by Sultani et al. [6]. The task of anomaly detection in video surveillance footage spans a wide variety of situations each with differing versions of normal and abnormal activity. The broad specification of the task has allowed for many papers to be written on the topic, each describing an architecture with a unique approach. The approach taken in the paper to be reimplemented is more of a high-level approach where the researcher is not as concerned with the fine-grained features appearing in frames but rather leaves these nuances to the network to learn.

Sultani et al. [6] focuses on creating opposing streams of footage using weakly-labeled training videos to separate footage into 'normal' or 'anomalous'. Labels only exist at video level and not frame level therefore the network is not told where exactly an anomaly in an anomalous video occurs. The network learns to predict an anomaly as well as avoid triggering false alarms which appear to be similar to anomalies. This is a crucial function of the framework made possible due to the video level labels provided by the dataset used for training.

The product backlog includes the steps to be taken to implement the architecture such that it produces results in line with the quality of results mentioned in the experiments of Sultani et al. [6]. The quality of the results will be determined by reproducing a selection of the experiments conducted by the researchers. Thereafter, the framework's functionality will be extended by classifying the environment as crowded/non-crowded and applying different processes based on the classification. The merit of these extensions will be validated by once again running the model on the benchmark experiments conducted by Sultani et al. [6] and comparing performance and results to those of the original model. Following the changes to the model, it will be applied to certain scenarios in attempt to achieve convincing proficiency in a sub-domain of anomaly detection. The backlog will include details on the systems and computing power required for training and testing of the model. This backlog will consider a trained model which implements the above mentioned work to be the final version of the product. The product may be extended and packaged after this step depending on time constraints.

# 2 Motivation

The applications of anomaly detection in the modern world are almost endless. Examples that quickly come to mind include automated CCTV monitoring and reporting, filtering of data streams and enabling of smarter robotic systems i.e self-driving vehicles. The architecture devised by Sultani et al. [6] holds the higher-level benefit of being relatively versatile as it does not fundamentally depend on visual feature recognition and object reconstruction but rather proposes a framework for learning from anomalies and anomalous-seeming, yet normal, activity. This makes it easier to adapt the framework to activities such as fraud detection, fault detection, system health monitoring, event detection in sensor networks, and detecting ecosystem disturbances. An application of anomaly detection which may potentially be implemented in the application phase of this project is the detection of vehicle hijackings by CCTV cameras and blacklisting the registration number of the vehicle. Applying the improved framework to this scenario will ideally allow the benefits of the crowded/uncrowded improvements to be displayed as car hijackings can occur in the midst of other vehicles (which will most likely create a change in the surrounding environment) or in empty parking lots (which will depend on motion prediction of the suspect for detection). This application can be further developed to include video enhance-

ment and facial detection functionality which ultimately provides society with a very useful tool in preventing/addressing such incidents.

# 3 Methodology

## 3.1 Outline

The scope of this project consists of three main tasks and an optional fourth task. The main tasks are: the implementation of the paper written by Sultani et al. [6] and verification of results achieved by them; the extension of functionality to the original framework which includes environment classification, peak window monitoring and motion prediction (this will include a significant amount of work in the testing phase); application of the modified framework to a real-world scenario and a review of the framework's performance in a sub-domain of anomaly detection (this phase will also require a data collection/preparation phase which may require a considerable amount of work). After the completion of the last mentioned phase, a trained model of the modified framework on a real-world scenario will be considered the end product of the backlog. Depending on the rate at which the project progresses, there may be an additional phase of packaging the application.

## 3.2 Dataset

The dataset to be used is provided by Charlotte Vision Laboratory and consists of 95.5 GB of CCTV video footage. The dataset can be summarized as containing long untrimmed surveillance videos which cover 13 real-world anomalies, including Abuse, Arrest, Arson, Assault, Accident, Burglary, Explosion, Fighting, Robbery, Shooting, Stealing, Shoplifting and Vandalism. The dataset was collected by training ten annotators (having different levels of computer vision expertise) to collect the dataset. Videos were searched on YouTube [1] and LiveLeak [3] using text search queries (with slight variations e.g. "car crash", "road accident") of each anomaly. The dataset can be stored and accessed locally but the machine it is stored on must take into consideration the computing power required which is mentioned in the following section. GPU processing is required in order to keep training and testing times reasonable and this may imply that work will be done on a remote server in which case the dataset will be stored there.

## 3.3 System Requirements

Taking into consideration the size of the dataset mentioned above, there will be a minimum specification of computing power required from the system training the model. Sultani et al. [6] split the dataset into 800 normal and 810 anomalous videos for training and 150 normal and 140 anomalous videos for testing. The distribution of video length is displayed in Figure 3 of Sultani et al. [6]. The same split will be followed in the reimplementation of the

framework in order to reproduce the results as closely as possible before progressing on to extensions of the framework.
Figure 8 of Sultani et al. [6] demonstrates that 8000 training iterations and upwards should be conducted before one can expect frame-level anomaly score predictions which resemble the ground truth of the video in focus. A single training iteration consists of the steps outlined in section 3.4. A function taking inputs of: the computation required for each step in an iteration, the number of iterations to be performed, the computing power of the system; and outputting the time required to train and test the model; will be established in order to determine the system required to efficiently develop the framework. The storage location of the dataset and caching abilities thereof will also be taken into consideration by the function.

### 3.4 Implement Framework and Visualization Tools

This is essentially the beginning phase of the project. A first step to be taken is to set up the environment in which the application will operate which includes the system requirements mentioned in the above section, an execution platform/environment, the libraries to be made available to the program during execution, a git repository to track progress and an access point to the dataset.
The libraries prescribed by Sultani et al. [6] include Keras version 1.1.0 [2], Theano 1.0.2 [7] and Python 3 [8] (running on Ubuntu 16.04 or later). A requirements.txt will be created in order to correctly set up the execution environment with the relevant software.
The details of implementation of the framework will now be covered. An outline of the model's processes can be seen in Figure 1, Sultani et al. [6]. From the pool of 800/810 normal/anomalous videos in the training set, a video is selected and divided into a fixed number of non-overlapping segments. A video which is labeled as anomalous will have its temporal segments filter into a positive bag (which is a collection of video instances coming from an anomalous video - note that not all of these instances will depict anomalous activity but at least one of them will). A video labeled as normal will have its temporal segments filtered into the negative bag (all instances depict normal activity). We randomly select 30 positive and 30 negative bags as a mini-batch. Applied to these instances within their bags is a C3D feature detector which extracts 3D convolution features. We use this feature representation due to its computational efficiency and the evident capability of capturing appearance and motion dynamics in video action recognition. Before computing features, we re-size each video frame to $240 \times 320$ pixels and fix the frame rate to 30 fps. We compute C3D features for every 16-frame video clip followed by normalization. To obtain features for a video segment, we take the average of all 16-frame clip features within that segment. We input these features (4096D) to a 3-layer fully connected neural network. The first fully connected layer has 512 units followed by 32 units and 1 unit fully connected layers. 60% dropout regularization is used between fully connected layers. We

use ReLU activation and Sigmoid activation for the first and the last fully connected layers respectively, and employ Adagrad optimizer with the initial learning rate of 0.001. We compute gradients by reverse mode automatic differentiation on computation graph using Theano. Then we compute loss as shown in Eq. 6 and Eq. 7 of [6] and back-propagate the loss for the whole batch. The trained model will operate on a provided video segment and assign it an anomaly score.

This model is to be accompanied by a basic interface displaying a graph of the anomaly scores along a section of video as well as the video beside the graph such that a user can view video segments and their corresponding to anomaly scores with ease. This aspect of the project will be implemented in the form of a simple web application using the Streamlit python library.

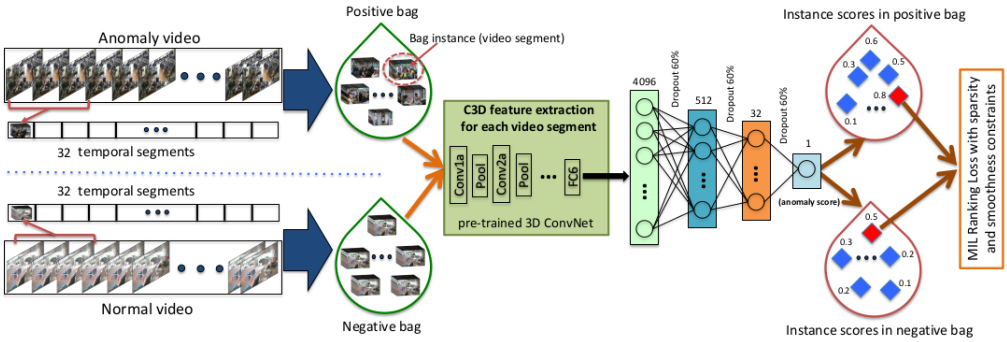


Figure 1: Deep MIL Framework applied to anomalous and normal bags of video segment instances. Given the positive (containing anomaly somewhere) and negative (containing no anomaly) videos, we divide each of them into multiple temporal video segments. Then, each video is represented as a bag and each 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 bag and the negative bag.

## 3.5 Benchmark Experiments

Due to the variation in anomalies, the task of testing an anomaly detection framework becomes challenging to automate and manual testing by verifying positives with human consent is a tedious process to repeat with each adjustment to the framework. We divide our dataset into two parts: the training set consisting of 800 normal and 810 anomalous videos and the testing set including the remaining 150 normal and 140 anomalous videos. Both training and testing

sets contain all 13 anomalies (mentioned in 3.2) at various temporal locations in the videos. The most efficient testing method seems to be to annotate videos with frame-level labels of anomalous activity and count an anomaly report given within the labelled area as a true positive and an anomaly report given outside of the labelled area as a false positive. Any anomaly which goes unseen is considered a false negative and each normal segment which is not reported is considered a true negative. This will form the confusion matrix which will be the main testing metric. Sultani et al. [6] mentions two additional state-of-the-art frameworks to compare results to. The implementation of these will be investigated and confusion matrices will be compared by plotting false positives against true positives and computing ROC (receiver operating characteristic) and AUC (area under curve) metrics for each framework.

### 3.6 Extend Framework to Classify Environments

In this phase of the project the framework's functionality will be extended in such a way that the problem domain is classified into crowded and uncrowded environments. This distinction allows different anomaly detection strategies to be applied when the scenario works in the favour of the particular strategy and thus we reduce the problem of broad anomaly detection into a simpler problem of identifying crowded environments and uncrowded environments. In order to reduce the penalty on the model's performance when introducing the environment classifier, the classifier will make use of as much of the work done by the original model as possible. Before finalizing the method behind the classifier, further research will be required during this phase of the project. A proposed method which will be convenient to apply to the original model will be a function which calculates the spatial difference between features from identical regions of the frame in consecutive temporal frames. This score can be calculated dynamically and, in the case that it exceeds/falls below a certain threshold, certain functionality of the model will be enabled/disabled. The proposed strategy should also be able to avoid the potential pitfall of classifying gradual changes in the environment (day to night) as activity indicative of a crowded/erratic environment. The details regarding which functionality is considered default and which functionality is specific will be determined by experimenting with the levels at which the algorithm's contributions become significant. The initial/default configuration of the model will be such that performance is maximized which will most likely result in the motion prediction (applied to uncrowded environments) being avoided when possible as it is far more computationally intensive than the peak window monitoring strategy which is essentially a calculation performed on the results of the original model rather than an extension of the actual model.

### 3.7 Implement Peak Window Monitoring on Crowded Environments

In the case that an environment's spatial difference score indicates that it is 'crowded' to a large enough extent, the model will disable the use of the motion

prediction and implement peak window monitoring. This will result in better performance of the model due to less computations required. The framework will be required to identify peaks in the anomaly score graph and calculate differences between the scores surrounding the peak for a length of time inversely proportional to the height of the peak. The peak will be identified by searching for anomaly scores above a threshold and waiting for a negative gradient in the graph thereafter. The difference in scores contained in the sliding windows will contribute to the diagnosis of the segment with a higher difference resulting in a higher likelihood of anomalous activity reported. The anomaly score graph mentioned under 3.4 will outline the regions to be used as windows in the case of a peak.

### 3.8 Implement Motion Prediction on Uncrowded Environments

In the case that an environment's spatial difference score indicates that it is 'uncrowded' to a large enough extent, the model will enable the use of the motion prediction disable peak window monitoring. This feature of the model operates on the assumption that static environments are unresponsive and will not show reactive features which may have been used to detect an anomaly. The framework will then rely on picking up subtleties in the motion of a familiar object i.e., the erratic movement of someone hijacking a vehicle in an empty parking lot will not be recognized by a model which has learnt the movements of pedestrians getting into their vehicles. The motion prediction sub network is to be adapted from a paper by Nguyen and Meunier [5] which implements a U-Net with skip connections. The motion stream is expected to associate typical motions to common objects while ignoring static background. The presence of skip connections allows high-level feature recognition to be related to low-level details. One can imagine that this is useful when predicting the instantaneous motion of an object. The ground truth of the optical flow is outputted by FlowNet2 [4]. The loss between the sub-network's outputted optical flow and the ground truth is measured by calculating the absolute distance between the optical flow given by FlowNet2 when provided with two consecutive frames; and the optical flow outputted by the sub-network when provided with only the first of the two frames given to FlowNet2. The differing input required by the optical flow versus the original model is evident early on (where the original model divides videos into segments and operates on each segment individually, the optical flow network requires a fine-grained approach as it needs to calculate the motion between two consecutive frames). This discrepancy makes it difficult to integrate the two models and may result in a completely separate model being set up for the optical flow. The optical flow network will require separate training during the training phase and will only be included in the testing phase when the environment classifier allows for it.

## 3.9 Benchmark Experiments and Comparison of Results

The same experiments conducted on the model tested in 3.5 will be conducted on the extended framework. These experiments result in a confusion matrix for the original version this framework, the extended version and two other state-of -the-art frameworks. The confusion matrices will be used to plot true positives versus false positives and the ROC and AUC will be calculated in each case. The results will be compared to those obtained in section 3.5 in order to see if the extensions have helped in the correct classification of difficult edge cases. These experiments should be done having already tested the capabilities of the environment classification algorithm, peak window monitoring and motion prediction in isolation.

## 3.10 Analysis of Extensions

An in-depth analysis of the performance of the extended framework will be conducted during this phase. The details of implementation of the extensions may be fine-tuned to reduce performance hits where contributions are insignificant and introduce functionality at an earlier environment difference score 3.6 if it seems it may be useful i.e., experimenting with the level of activity at which an environment diagnosis switches from crowded to uncrowded. Finally, the merit of the changes made to the framework will be considered and a decision will be made on which changes will be enforced/discarded.

## 3.11 Real-World Scenario

In this phase of the project, a real-world scenario with potential to show a range of anomalies within a sub-domain will be identified for testing of the framework. The selected scenario should allow for crowded and uncrowded environments to be tested in order to challenge the model to use different aspects of functionality. The scenario of vehicle hijackings is a potential choice as it would be a useful application for society and, provided a decent dataset can be obtained, should lend various environments with differing levels of background activity. Another scenario to be considered is road accidents for similar reasons to those of vehicle hijackings. The idea is to train the model for the specific scenario and observe the level of ease at which the framework classifies anomalous activity when the anomalies are relatively similar to each other but the environment varies drastically. This will demonstrate whether the model can be useful in the context of a real-world application. A confusion matrix will be set up to observe the accuracy of classifications. This phase will also be used to gain any insight into potential shortcomings of the newly introduced functionality.

## 3.12 Data Collection and Training

This phase will be completed in tandem with the previous section as the selected choice of application will greatly depend on the availability of data for the

scenario. The dataset used by Sultani et al. [6] does not mention any data on vehicle hijackings but does contain 150 instances of road accidents. Further searching for data of these instances will be carried out during this phase and the decision as to which scenario is selected will depend on the results of this phase. An 80/20 split will be used for training/testing of the model.

### 3.13 Review of Application

An analysis of the performance of the framework on the chosen application will be conducted during this phase of the project. The model should ideally achieve an accuracy of above 85% in summary of the confusion matrix if it is to be considered a useful application for society. The successes and shortcomings of the model's performance will be discussed and linked back to technical decisions made earlier on in the implementation.

### 3.14 Packaging of Application

Provided the previous phase of work yields positive results and the timeline of the project is adhered to, there may be an opportunity to package the framework as a trained model to enable convenient use of the framework. This will be explored during this phase and may require more of a software engineering approach. The project is, to a large extent, considered a research project and therefore this is not a mandatory aspect of final result.

### 3.15 Conclusion and Finalizing of Report

The project's work will be summarized and reflected upon including a description of the state that the project is left in. Key findings will be highlighted and the potential for further work will be discussed. The project concludes with the finalizing of the report and git repository.

## References


[1] J. E. Burgess. Youtube, 2011.

[2] F. Chollet et al. Keras, 2015. URL `https://github.com/fchollet/keras`.

[3] F. Çömlekçi and S. Güney. An alternative media experience: Liveleak. In *International Conference of Design, User Experience, and Usability*, pages 62–70. Springer, 2014.

[4] T.-W. Hui, X. Tang, and C. C. Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. pages 8981–8989, 2018.

[5] T.-N. Nguyen and J. Meunier. Anomaly detection in video sequence with appearance-motion correspondence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1273–1283, 2019.

[6] W. Sultani, C. Chen, and M. Shah. Real-world anomaly detection in surveillance videos. pages 6479–6488, 2018.

[7] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

[8] G. Van Rossum and F. L. Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.