Violent Behaviour Detection
Using Time-Sequence Pose Estimation Data

Sean Farrugia

Supervisor: Mr Chris Farrugia

June, 2022

A dissertation submitted to the Institute of Information and Communication Technology in partial fulfillment of the requirements for the degree of B.Sc. (Hons.) Multimedia Software Development

Authorship Statement

This dissertation is based on the results of research carried out by myself, is my own composition, and has not been previously presented for any other certified or uncertified qualification.

The research was carried out under the supervision of Mr Chris Farrugia.

Sean Farrugia

June 5, 2022

Copyright Statement

Sean Farrugia

June 5, 2022

Acknowledgements

I would like to express my gratitude and appreciation to my family and friends for their encouragement and unwavering support throughout.

I want to convey my heartfelt appreciation to my mother. She encouraged me to continue my studies and submit my dissertation proposal during trying times; advice I'm pleased I followed.

I would also like to give my thanks to my dissertation supervisor for his guidance.

Table of Contents

List of Abbreviations

AI Artificial Intelligence. 2, 5

ANN Artificial Neural Network. v, ix, 9, 10, 11, 12

CNN Convolutional Neural Network. v, vi, ix, x, xi, 1, 2, 4, 11, 12, 15, 17, 18, 25, 32, 33, 34, 35, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57, 58, 60

LSTM Long Short-Term Memory. 1, 33, 48, 54, 57

ML Machine Learning. v, 1, 3, 4, 5, 6, 9, 12, 15

PAF Part Affinity Fields. 14, 27, 42

RBF Radial Basis Function. 19

RF Random Forest. 17, 59

RNN Recurrent Neural Network. 17

SVM Support Vector Machine. v, vi, ix, 1, 4, 6, 7, 17, 18, 19, 21, 25, 35, 46, 47, 48, 54, 58, 59, 60

ViF Violent Flow. ix, 18, 19

YOLO You Only Look Once. vi, x, 1, 25, 28, 29, 36, 37, 38, 39, 58

List of Figures

List of Tables

Abstract

Despite the increase of security cameras, Violent crime is still a global challenge as manually analysing the footage has drawbacks. The aim of this study is to make use of Machine Learning (ML) and Computer Vision to train a classifier in detecting violent behaviour using time-sequence data collected from a standard camera's video clip. Posture data was extracted by utilising OpenPose Pose Estimation, You Only Look Once (YOLO) Object Detection and OpenCV Tracking API. A keypoints dataset was proposed to train a Convolutional Neural Network (CNN) and a Support Vector Machine (SVM) to classify whether the time-sequence posture data is 'Violent' or 'Non-Violent'. Python was utilised to use Tensorflow through Keras to build the architecture of the CNN, train the model and test it. The CNN architecture includes 1DConv and Long Short-Term Memory (LSTM) layers due to their potency with time-sequence data and Dropout layers to prevent overfitting. SciKit-Learn was used to train and test the SVM with a linear kernel. The proposed CNN model achieved the most promising results with 79% accuracy on the testing set.

Chapter 1

Introduction

According to Cheng et al. (2020), although the number of surveillance cameras in public places has increased, manually monitoring them for any activity remains a problem; this is due to expenses and the time it takes for observers to identify a crime and respond. Violent crime is a worldwide challenge that has an impact on the victims' quality of life as the criminal employs hostile force against them and at times the victim's life is placed at risk. The demand for a real-time automated monitoring system is increasing. Video footage taken from any device may be automatically, and in real-time, evaluated by Artificial Intelligence (AI) to aid with emergency calls in a timely manner, which is far more efficient than manual surveillance (Ramzan et al., 2019).

1.1 Motivation

This study was conducted due to a personal interest in Computer Vision and the ever growing range of techniques in this area. This technology was used numerous of times beforehand for personal projects namely for Traditional Painting Recognition using SIFT, and Classifying Swords with a Convolutional Neural Network (CNN).

The theme of action and violence is a personal interest as well; namely, as mentioned beforehand, weapons and swords but also movies and video games such as fighting games, first-person shooters and action role-playing games. However, in real life it is saddening hearing news about individuals getting involved in violent activities, where in some cases people lose lives. This may be due to aid not arriving earlier or no one reports the incident.

Therefore it seemed ideal to combine both interests, alongside acquired skills throughout the degree, to research a method that automatically detects violent behaviour, which could inform surveillance as quickly as possible.

1.2 Aims and Objectives

The hypothesis of this research is: by making use of Machine Learning (ML), it is possible to train a classifier to detect violent behaviour with time-sequence pose estimation data collected from a standard camera's video clip. Hence, the aim of this study is to develop a system that can identify violent behaviour as fast as possible by classifying postures of individuals. The objectives are:

- To develop a custom dataset that follows common practices in posture recognition
- Review existing algorithms and propose a suitable model
- Predict postures as quickly as feasibly possible and analyse the results attained

From those objectives, the following research questions were developed:

1. What dataset is required to train the models into recognizing violent behaviour?
2. What algorithm can be utilized to detect violent behaviour?
3. How accurately can the proposed prototype detect violent behaviour?

1.3 Chapters Outline

In this section each chapter in this document is described.

1.3.1 Chapter 2: Literature Review

Conducted secondary research is discussed on Machine Learning (ML), classifiers such as Convolutional Neural Network (CNN) and Support Vector Machine (SVM), Computer Vision, Pose Estimation, Violent Detection approaches using Computer Vision and datasets utilised for Violent Behaviour Detection.

1.3.2 Chapter 3: Research Methodology

The prototype implemented based on the research made in Chapter 2 is described in Chapter 3, alongside specific techniques involved. How primary data was acquired is also discussed in this chapter.

1.3.3 Chapter 4: Analysis of Results and Discussion

The performance of the implementation is thoroughly evaluated and discussed in Chapter 4. In addition, the results and findings of the tests conducted are also described.

1.3.4 Chapter 5: Conclusions and Recommendations

The conclusion of the research is found in this chapter, where the research results are summarized and future work is recommended.

Chapter 2

Literature Review

2.1 Introduction

Violent crime is an international issue, affecting the quality of life of victims involved as the perpetrator uses or threatens to use hostile force upon them, sometimes even involving weapons. Violent crimes are committed everyday, according to Cheng et al. (2020), the number of surveillance cameras in public places continuously increases, however monitoring them manually deems to be expensive. The need and interest in a real time automated monitoring system grows. By making use of Artificial Intelligence (AI), a simple footage from any device can be analysed automatically and in real-time to assist in emergency calls in a timely manner that beats manual observation (Ramzan et al., 2019).

2.2 Machine Learning (ML)

Machine Learning (ML) is a subset of AI. According to both studies conducted by Jordan & Mitchell (2015) and Carleo et al. (2019), ML is a field of study that is rapidly advancing and impacting various other scientific domains. Furthermore, these authors have highlighted how this area has influenced fields such as health-care, speech recognition, manufacturing, finance and others. Jordan & Mitchell (2015) explained how ML addresses the challenge of a computer comprehending data and recognising patterns through experience and self-learning. In fact, Carleo et al. (2019), brings forth an example as how ML is implemented in a self-driving car and how from the data that it receives, it formulates a plan of action which preferably is the safest; this is possible because it has previously learnt what is harmful and what is not.

2.3 Learning Paradigms

According to O'Shea & Nash (2015), self-learning in ML consists of primarily two approaches: Unsupervised and Supervised Learning. In Unsupervised Learning, the dataset presented to the model during the learning process is not labelled; the patterns and data are identified by the model on its own. This increases the unpredictability of the results, but it could be useful for models that require complicated task processing. The latter method, the Supervised Learning, decreases unpredictability by labelling the information inputted to the model from the dataset. During the self-learning process, the input label is compared to the predicted output, and the model adjusts itself accordingly. As a result, the model has less potential error, which explains why this approach is preferred (O'Shea & Nash, 2015).

2.4 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised ML approach for classifying between two classes. According to Noble (2006), the training data is plotted on a plane that is represented by data points during the fitting process. The co-ordinates of these points are determined based on the features of the data. Afterward, as shown in Figure 2.1, the SVM divides the plane into two sides by drawing a line, termed a 'Separating Hyperplane', which separates the points into two clusters.

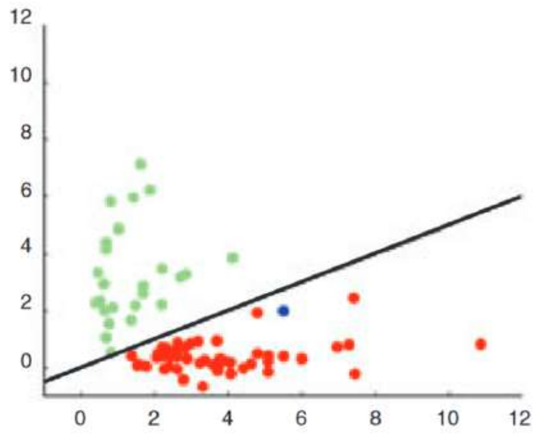

Figure 2.1: A visual representation of an SVM plane, the data points of the two categories coloured differently and being separated by the 'Separating-Hyperplane' (Noble, 2006).

Pisner & Schnyer (2020) further discusses how the 'Separating Hyperplane' is optimized by using the 'Maximum-Margin Hyperplane' theorem. As seen in Figure 2.2, this approach positions the line precisely in the middle of the two clusters, where the nearest data points of each class are at maximum distance. When new data is predicted, it is categorised by the SVM by assessing on which side of the plan it corresponds to based on its features.

However, Noble (2006) pointed out that some data points are plotted on the opposite side of the plane as seen in Figure 2.3. In such scenarios, A 'Soft Margin' is added to the SVM algorithm. This permits some data to be pushed from their side without modifying the separating hyperplane, resulting in fewer inaccurate classifications overall. For this to be applied, a parameter controlling how many points are permitted to cross the margin would be defined beforehand.

However, even with a 'Soft Margin', complicated datasets would be far too difficult to

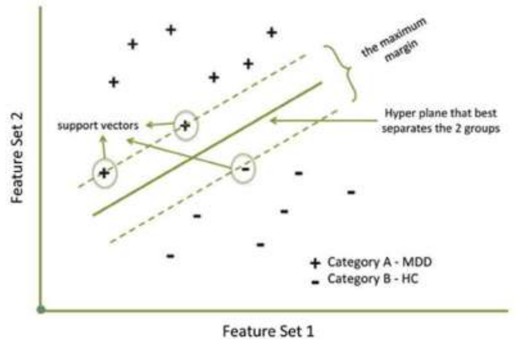

Figure 2.2: The Hyperplane using the Maximum-Margin Hyperplane theorem (Pisner & Schnyer, 2020).

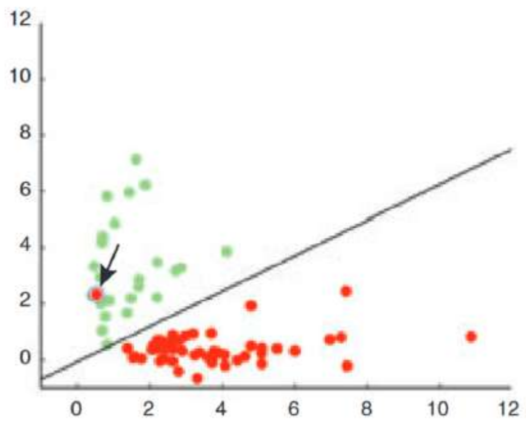

Figure 2.3: Soft Margin allowing an error in the dataset which results in a data point to be on the opposite side of the plane (Noble, 2006).

separate with a straight line when plotted with their associated features and categories, as shown in Figure 2.4. According to Pisner & Schnyer (2020), a non-linear classifier using a curved hyperplane is necessary in such scenarios. A kernel approach is used to add a dimension to the plane, transforming it from 2D to 3D. This will modify the orientation of the plane, making it possible to separate the data linearly as seen in Figure 2.5.

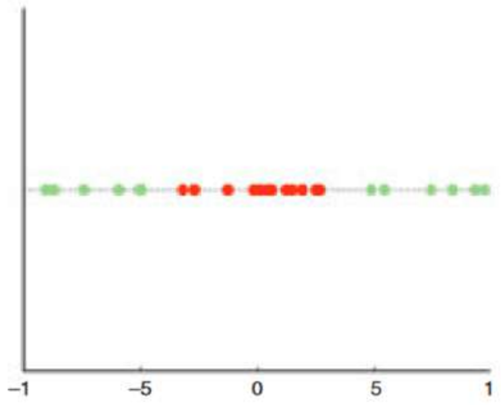

Figure 2.4: Data points that are linearly inseparable (Noble, 2006).

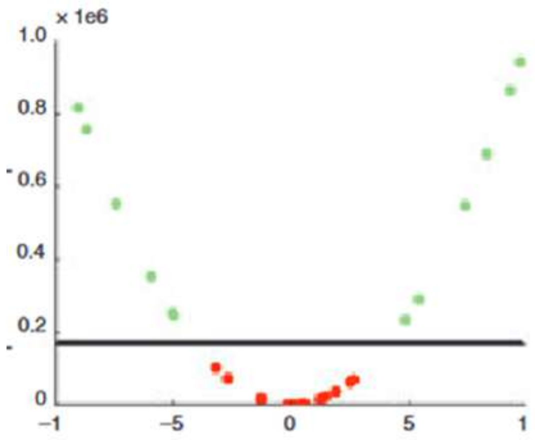

Figure 2.5: Due to the increase of dimensionality and the orientation of the plane, it is now linearly separable (Noble, 2006).

2.5 Artificial Neural Network (ANN)

Artificial Neural Network (ANN) is another type of ML classifier. ANNs are computer processing systems that mimic a biological nervous system. They are made up of numerous computational nodes called neurons that are arranged in layers (O'Shea & Nash, 2015). An input is processed by these layers and a predicted output is produced based on a set of weights that are constantly tweaked during the prediction process. The classification of the input is determined by the heavily weighted prediction. The usual structure of an ANN, as shown in Figure 2.6, consists of three type of layers: Input Layer, Hidden Layers, and Output Layer.

- The Input Layer is given a multidimensional vector to be processed by the subsequent layers (O'Shea & Nash, 2015).
- The Hidden Layers are typically a collection consisting of multiple sequential layers. Each one will process the data received from the preceding layer. The weights of the prediction are gradually altered by the layer based on the received data to improve the classification. Once finished, the data is then passed to the subsequent hidden layer, that repeats the process (O'Shea & Nash, 2015).
- The Output layer is provided the finally weights of the Hidden Layers, from which a prediction is made and outputted (O'Shea & Nash, 2015). Through self-learning, the output could be utilised to improve the Hidden Layer's prediction by comparing it with the actual result. This process is known as 'Back Propagation', where the predicted result is sent back to the Hidden Layers to adjust the weighting process (O'Shea & Nash, 2015).

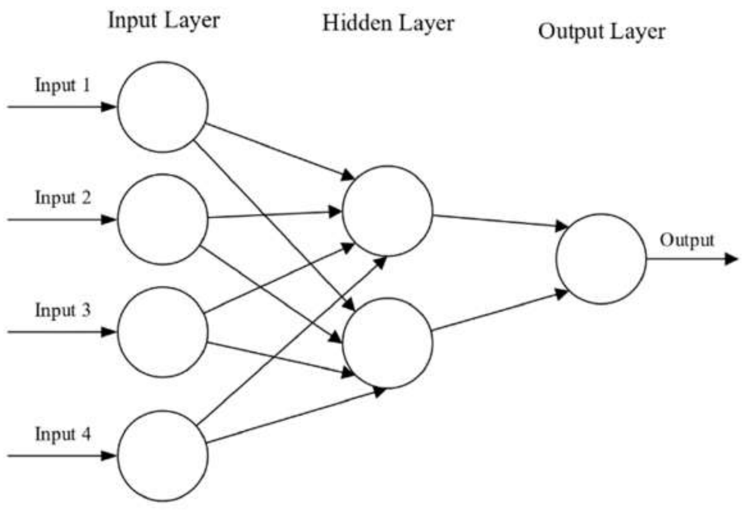

Figure 2.6: A simple ANN structure showing all 3 types of Layers (O'Shea & Nash, 2015).

2.6 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a variant of ANN. Similarly, it is composed of layers and neurons and go through a self-learning process to improve. However, CNNs differ in their architecture as they could be designed specifically to process imagery and are typically favoured in the field of image pattern recognition. This is due to their efficiency in evaluating images as the number of processes needed are reduced when compared to an ANN (Albawi et al., 2017). Thus, as the image gets larger, ANN's drawbacks become more evident (O'Shea & Nash, 2015). Albawi et al. (2017) claim that this allows researchers to handle more complex tasks with larger models, which are increasingly challenging with ANNs.

CNNs could also learn features that are spatially independent. As a result, a feature could be located anywhere in the inputted image, as the CNN model will detect them regardless (Albawi et al., 2017). The authors also discuss how in CNNs, different features are extracted in different layers of the model. In most cases, the models begin by extracting basic characteristics such as edges in the initial layers, then go on to simple shapes in subsequent layers, and lastly to higher level features in the final layers.

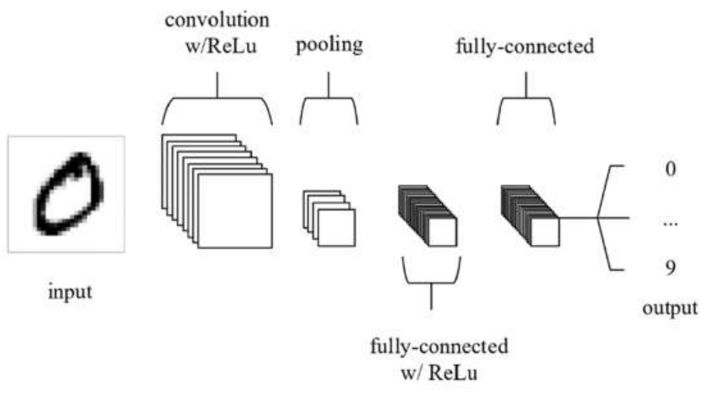

Figure 2.7: A simple CNN structure showing all types of layers (O'Shea & Nash, 2015).

As seen in Figure 2.7, the CNN layers are categorised into four groups: the Input Layer, the Convolution Layers, the Pooling Layers and the Fully-Connected Layers.

- The Input Layer generally uses an order 3 tensor as an input, meaning three values: the columns and rows of the image, as well as the number of channels (Wu, 2017).
- The Convolution Layer extracts information from the input and determines the output of the neurons (O'Shea & Nash, 2015).
- The Pooling Layer downsamples the input to lower the number of parameters and dimensionality (O'Shea & Nash, 2015). This minimizes the level of detail extracted while also reducing the complexity for subsequent layers (Albawi et al., 2017).
- The Fully-Connected layer is used to arrange and condense the neurons. It could also be utilized at the end of the CNN model for classification, just like the Output layer of an ANN, as it weights the predictions made so far by the CNN (Albawi et al., 2017; O'Shea & Nash, 2015).

2.7 Computer Vision

Computer Vision is a subset of Machine Learning (ML) that deals with how a machine perceives information presented in visual or digital media. According to Szeliski (2010), this field has seen improvement in recent years, and it is now possible to train a model to track objects or human movement even when complex backgrounds are present in the digital media. This could be accomplished by building a model, and through self-learning, it is trained to recognise patterns within the image (O'Shea & Nash, 2015; Szeliski, 2010). The training procedure necessitates a dataset comprising a large number of data samples, categorised into different classes, from which the model could learn.

2.8 Human Pose Estimation

Human Pose Estimation is a computer vision technique that seeks to determine the body's keypoints or joints from a given image, such as shoulders, elbows, and wrists, . From these joints, an articulated human skeleton is created by the model as seen in Figure 2.8. This human skeleton could be generated in 2D or 3D (Munea et al., 2020). Furthermore, individual poses or Multi-Human poses in a single image are possible depending on the model used (Munea et al., 2020). Pose Estimation is categorised into two different approaches: Top-Down and Bottom-Up (Cao et al., 2017).

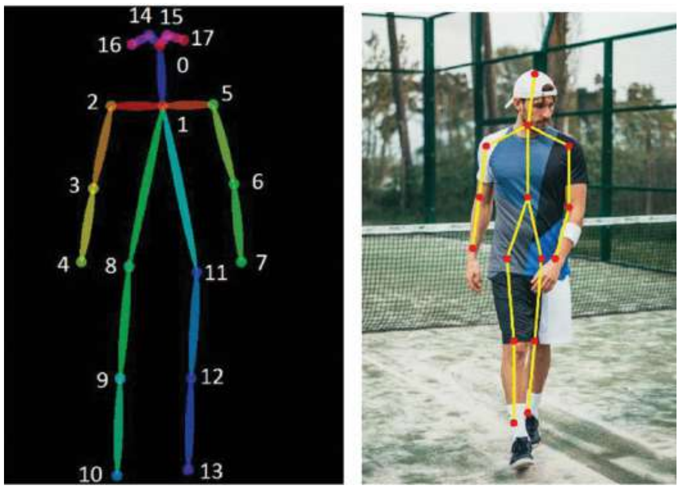

Figure 2.8: Human Pose Estimation applied on a picture, detecting the keypoints and an articulated human skeleton (Munea et al., 2020).

In the Top-Down approach, first the bounding boxes of the detected humans are calculated by an object detector. Following that, the pose of each human is detected by using a single-person pose estimator (Cao et al., 2017) as seen in Figure 2.9. According to Cao et al. (2017), this approach suffers from early commitment because if the individual is not detected, the pose estimator will not be applied. Furthermore, the computational cost is proportionate to the number of individuals in the image (Cao et al., 2017). In a study conducted by Papandreou et al. (2017), a technique that uses the Top-Down approach was proposed, named 'PoseNet'.

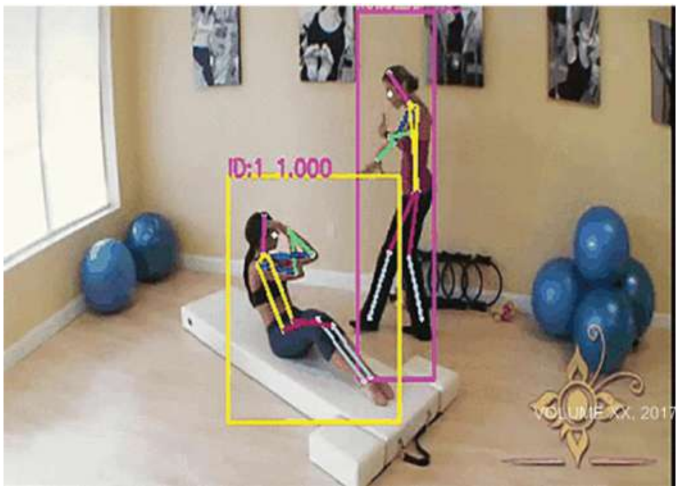

Figure 2.9: Top-Down Pose Estimation approach showing the Boundary Box and the skeleton generated (Munea et al., 2020).

The Bottom-Up approach, proposed by Cao et al. (2017), works in the opposite direction. The authors proposed OpenPose, a 2D pose estimation model utilising this approach. In Figure 2.10, the pipeline of OpenPose when estimating a posture is depicted. First, a heat map is generated, which represents the degree of certainty that a specific keypoint is situated at a given pixel. The main keypoints are identified using this map. Then these keypoints are grouped together into a single skeleton representation using Part Affinity Fields (PAF). PAF assesses the location and orientation of body limbs and finds any correlation between them regarding if they belong to the same human, as shown in Figure 2.10. If that is the case, they are regarded to be associated with the same body and thus a posture is developed from the keypoints.

Cao et al. (2017) states that the early commitment mentioned in the Top-Down approach is eliminated with this technique. In addition, the number of individuals detected in the image has shown to not hinder the run-time performance. Albeit the model tends to fail with overlapping or uncommon poses, as this is a fundamental Pose Estimation challenge, PAF has demonstrated that it aids in extracting more accurate poses in such instances (Cao et al., 2017).

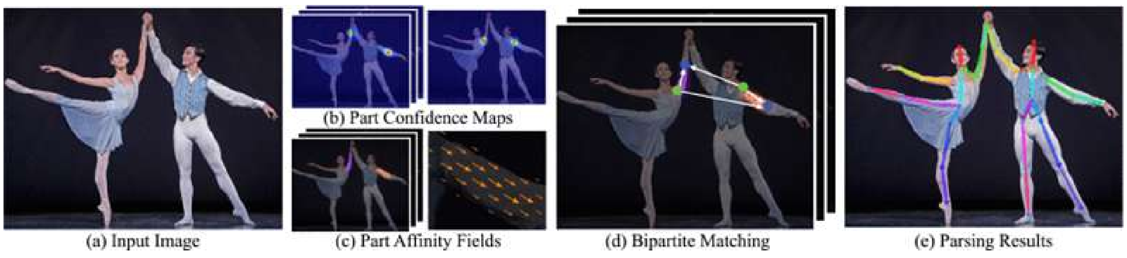

Figure 2.10: The pipeline of the Bottom-Up Approach, OpenPose, proposed by Cao et al. (2017).

2.8.1 Behaviour Recognition with Pose Estimation

Both of the work done by Gatt et al. (2019) and Vyas et al. (2019) concluded that the extracted human posture data, alongside ML and CNNs, may be used to develop a system that recognizes activities. Moreover, the data can be extracted from video clips taken from a normal camera.

The approach practiced by Gatt et al. (2019) to detect abnormal behaviour involved extracting posture data from subsequent frames and processing them into time-sequence data of shape (30, 34); the shape represents 30 time-steps and 34 features. The authors pointed out that Pose Estimation can return missing keypoints, however in such cases, the keypoints were filled in with a value of (0, 0) as was done in the research conducted by Xu et al. (2018). The data was then normalised to aid in the fitting process of a CNN with 1DConv and LSTM layers. This is because according to the authors, these have proven to work well with extracting features from time-sequence data. A patience callback was utilised with a value of 5 to prevent overfitting, achieving F1-score of 0.93.

The approach proposed by Vyas et al. (2019), to detect Autism Spectrum Disorder symptoms, also involved time-sequence pose estimation data. Unlike the research conducted by Gatt et al. (2019), the missing keypoints were filled out by applying a non-linear interpolation algorithm to predict them. Moreover, a PoTion representation of the data was generated instead of using the keypoints. A PoTion representation is a single RGB image that captures the changes of posture keypoints over a time-period as seen in Figure 2.11. The image contained data from approximately 301 frames or 10 seconds. The CNN architecture utilised Conv2D layers due to their potency with extracting features from an RGB image. The model achieved 72.4% Accuracy on the testing set.

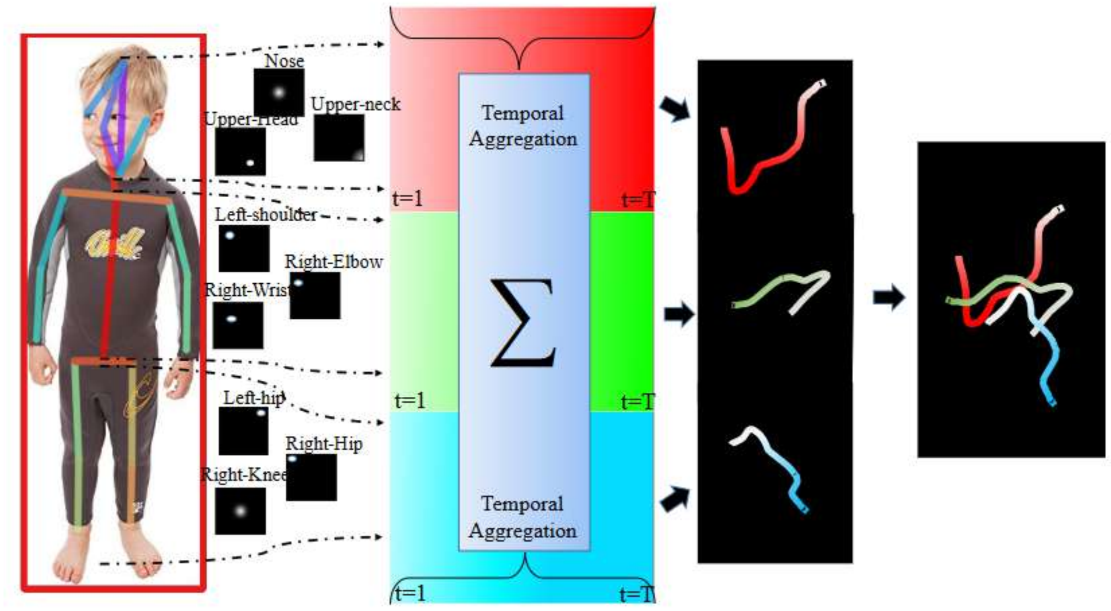

Figure 2.11: The process of generating a PoTion representation from the posture data (Vyas et al., 2019).

2.9 Violent Detection Using Computer Vision

In recent years, Violent Detection has received a lot of attention (Ramzan et al., 2019). Real-time violent detection involves a video rather than a static image, hence time is an valuable resource since a video is a sequence of frames. Therefore, the appropriate networks and algorithms must be employed to correctly recognise the behaviour of individuals. Various techniques were employed as discussed by Ramzan et al. (2019), ranging from analysing Motion Blobs (Gracia et al., 2015; Serrano et al., 2018), Optical Flow Vectors (Arceda et al., 2016; Fu et al., 2017; Xie et al., 2016), using keypoints from Pose Estimation (Nar et al., 2016; Nova et al., 2018) or even audio information (Mu et al., 2016) for feature extraction. As classifiers, several studies used Support Vector Machine (SVM), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Rule-Based Classification, AdaBoost or Random Forest (RF); some also employed a mix of these methods.

2.9.1 Motion Blob approaches

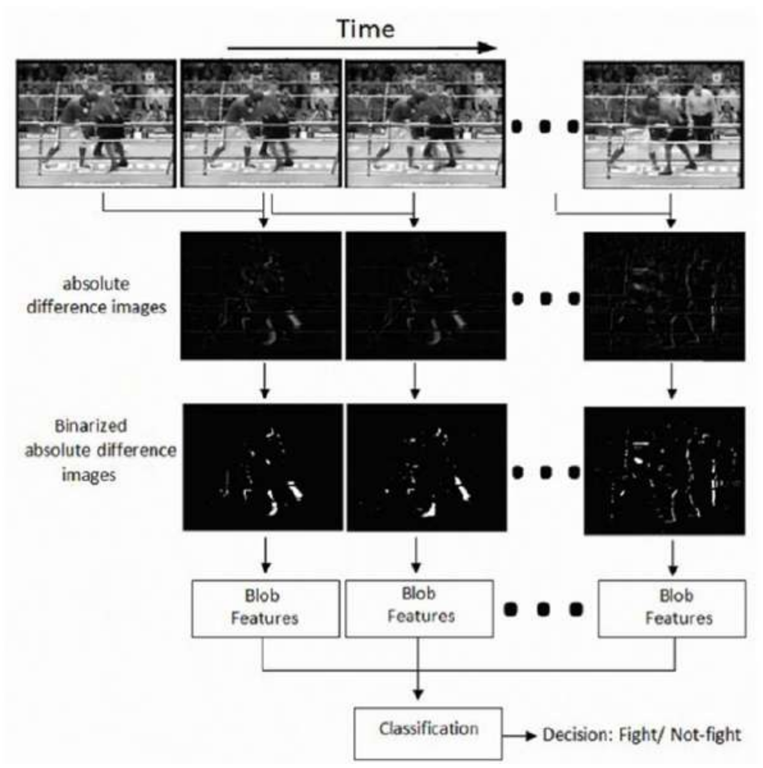
Time
absolute difference images
Binarized absolute difference images
Blob Features
Blob Features
Blob Features
Classification
Decision: Fight/ Not-fight

Figure 2.12: General diagram of the proposed method using Motion Blobs for classification (Gracia et al., 2015).

As previously stated, a video with consecutive frames is usually involved in violent detection. As shown in Figure 2.12, with the Motion Blob approach, a difference map between consecutive frames is generated to extract features, which are blobs of motion. According to Gracia et al. (2015), violent blobs will have a specific shape due to the velocity of the motion. In their study the difference map was converted to a binarized image, and only K largest blobs were utilised to train an SVM classifier; K being a given number. However, according to the author, this strategy was outperformed by state-of-the-art approaches, with accuracy ranging between 70% and 90%.

Instead of binarizing the image, the method employed by Serrano et al. (2018), creates a Motion History image but the effect of irrelevant parts such as background and noise are reduced, as shown in Figure 2.13. The author used a variety of classifiers but claimed that CNNs outperformed other classifiers by reaching accuracy of between 84% and 96%.

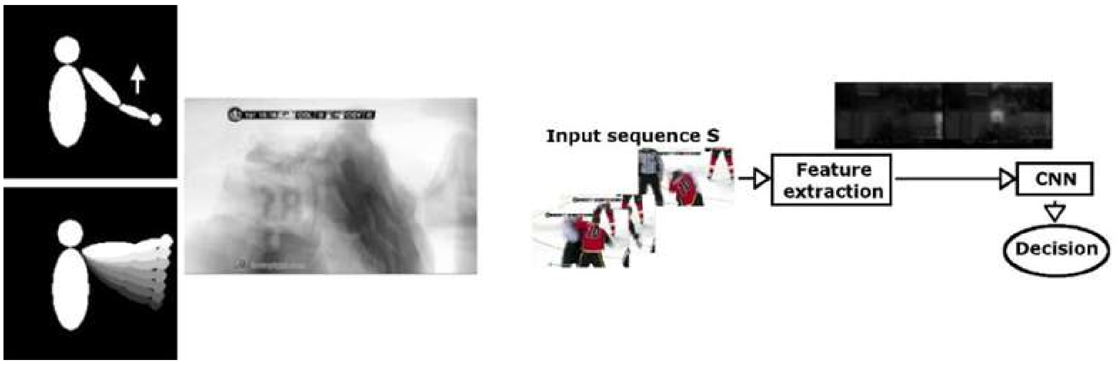

Figure 2.13: Example input and the resulted Motion History Image with the proposed method pipeline (Serrano et al., 2018).

2.9.2 Optical Flow Vectors approaches

The sequence of frames in a video can also be used to extract the trajectory and intensity of motion regions called Optical Flow Vectors. Optical Flow Vectors, like Motion Blob, will have a unique intensity and trajectory in a fight. For frames in sequence, a magnitude-change significance map is generated, illustrating the direction and magnitude of the pixel movement. In the method proposed by Arceda et al. (2016), Optical Flow Vectors were extracted by using the Violent Flow (ViF) descriptor with Horn-Schunck for violent scene detection, as seen in Figure 2.14.

Their method entailed generating a map that represented all of the frames in the clip and

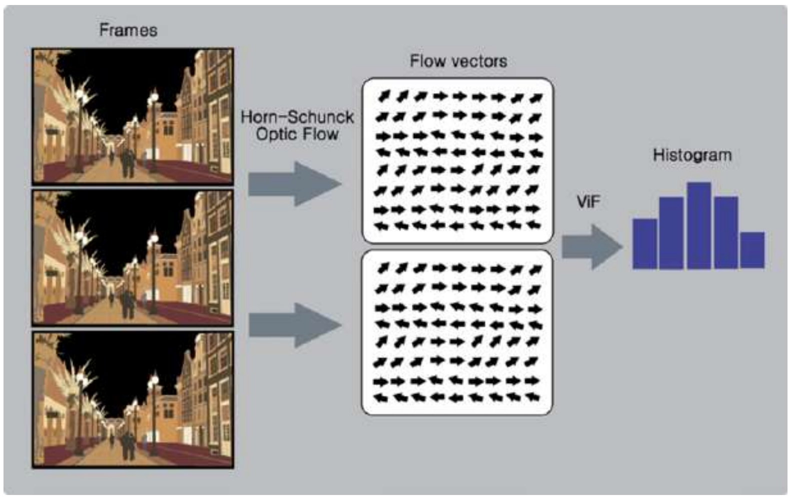

Figure 2.14: ViF descriptor in video (Arceda et al., 2016).

contained mean magnitude-changes for each pixel. The mean magnitude-change map is passed to a SVM classifier with polynomial kernel.

Similarly, the method proposed by Xie et al. (2016) utilised Optical Flow Vectors, which where normalised; then a 3D Histogram was generated. The histograms were fed to an SVM with a Radial Basis Function (RBF) for training and classification.

In the research conducted by Fu et al. (2017), an Optical Flow Image was constructed using Optical Flow Vectors, as seen in Figure 2.15. The Optical Flow Image depicts the motion trajectory from successive frames, with each direction represented by a different color. In their approach, the noise was removed from the Optical Flow Image, leaving only an image representing the violent movement that occurred. The feature extraction process was utilised to yield valuable data such as motion magnitude, motion acceleration and motion attraction. This is then passed into an SVM classifier for training and prediction achieving 78.6% accuracy (Fu et al., 2017).

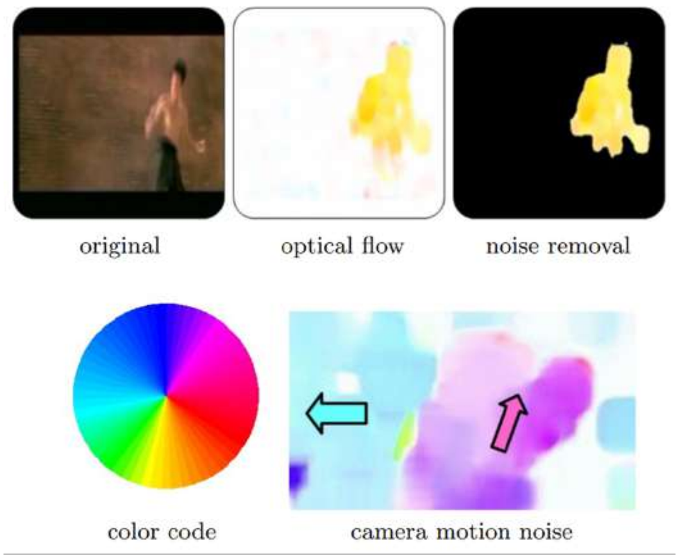

Figure 2.15: The process of converting an original image to a noiseless optical flow image representation of the violent motions. The figure also shows how the colour and the trajectory of the motion are related (Fu et al., 2017).

2.9.3 Pose Estimation approaches

As discussed in Section 2.8, Pose Estimation is a technique for extracting keypoints or joints from a human and creating a skeleton representation of the individual. The study conducted by Nar et al. (2016), used XBOX's Kinect 3D camera to estimate posture and extract keypoints positions. The extracted keypoints are then used to calculate 10 joint angles which are saved in a text file. The dataset, consisting of 100 of these entries, was processed using the Gradient Descent Method for classification, achieving around 85% to 91% accuracy. However, rather than humans, this study focused on a single person acting suspiciously, by performing specific postures at a bank ATM.

In a study carried by Nova et al. (2018), OpenPose was used for Pose Estimation. By using Human-Segmentation, the authors used physical contact between two-people, as seen in Figure 2.16, as an indicator that violent behaviour is present. A challenge met during the study was that OpenPose could not distinguish between two individuals. The order of the postures outputted varies across frames, making it challenging to follow an

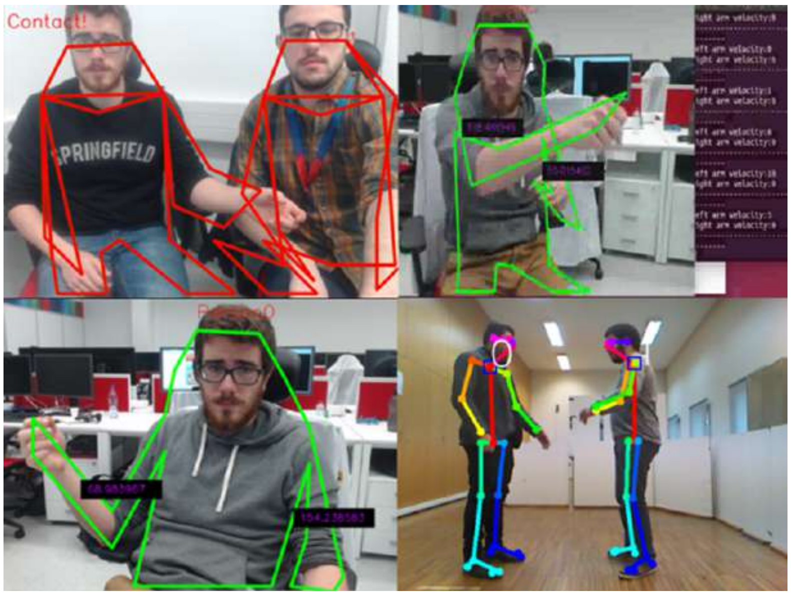

Figure 2.16: Examples of the features extracted: contact detection (top left), velocity (top right), angles (bottom left), and tracking (bottom right) (Nova et al., 2018).

individual. As a solution, Nova et al. (2018) proposed a tracking algorithm using Kernelized Correlation Filters, where a Region of Interest is established on the throat joint. After the keypoints were extracted the velocity and angle of the joints were calculated and saved in a CSV file. The dataset was then fed into the SVM classifier, obtaining 89% accuracy (Nova et al., 2018). The authors do note out, however, that their prototype is limited to two people and that touch must be detected.

2.10 Violent Behaviour Datasets

For a classifier to be trained, an adequate dataset must be used as it is decisive to the performance of the model; ideally being large and containing high-quality varied data. The datasets that are typically used in violent behaviour detection are described in this section.

The Hockey Fight Dataset, proposed by Bermejo Nievas et al. (2011) and illustrated in Figure 2.17, was introduced specifically for violent detection. The authors claimed that at the time of the study, publicly available datasets for action recognition were insufficient because they only covered simple actions. This dataset contains 1000 clips of hockey games categorised as ’fight’ or ’non-fight’, with each clip including 50 frames and 720x576 pixels.

According to Bermejo Nievas et al. (2011), the Hockey Fight Dataset, despite its diversity in fight patterns, it is restricted to specific scenarios. Therefore, a second dataset was proposed by the authors, the Movie Fight Dataset, illustrated in Figure 2.18. It includes 200 action movies video clips in various resolutions and environments.

Figure 2.17: Samples from the Hockey Fight Dataset (Bermejo Nievas et al., 2011).

Figure 2.18: Samples from the Move Fight Dataset (Bermejo Nievas et al., 2011).

The Violent-Flows Dataset, proposed by Hassner et al. (2012) and shown in Figure 2.19, is another rich dataset. This dataset was created to accommodate recognition of crowd behaviour for violent behaviour detection. It comprises of 878 clips, each of which has been scaled to 320x240 pixels and are of varied lengths. The samples are split into two classes: 'Violent' and 'Non-Violent'.

Figure 2.19: Samples from the Violent-Flows Dataset (Calzavara, 2020).

The above mentioned datasets are identified as benchmark datasets by Calzavara (2020) due to their high-quality and sufficient data, and are split evenly between fight and non-fight clips. However, Cheng et al. (2020) argues against this, pointing out that these datasets have limitations such as low image quality and videos with unrealistic or uncommon scenarios. Therefore, with the aim to solve these issues, a dataset called the RWF-2000, illustrated in Figure 2.20, was proposed by the authors. It consists of 2000 trimmed video clips of surveillance cameras in real-world scenarios, split into 'Violence' and 'Non-Violence' classes. The authors claim that this makes it the largest video dataset of violent detection.

Bianculli et al. (2020), argued that the RWF-2000, albeit being a large, high quality dataset, it is limited in preventing false positives. They discuss how the dataset lacked non-violent fast motions like hugs and high-fives. To address the listed issues, the authors proposed a dataset consisting of 350 high quality clips with a resolution of 1920x1080, labelled 'Violent' or 'Non-Violent'. It includes clips featuring violent and non-violent activity, as well as rapid non-violent actions including greetings, embraces, handshakes and high-fives, as seen in Figure 2.21.

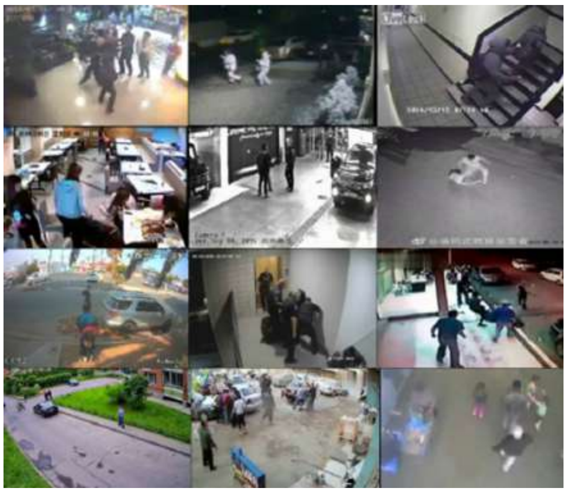

Figure 2.20: Samples from the RWF-2000 dataset (Cheng et al., 2020).

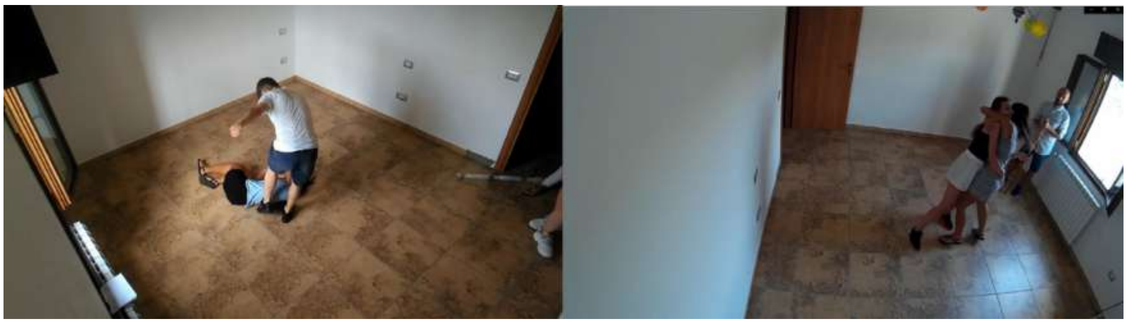

Figure 2.21: Samples from the dataset proposed by Bianculli et al. (2020); Example of Violent Clip (Left); Example of rapid non-violent action (Right).

2.11 Conclusion

In this chapter, secondary research was evaluated in order to develop a summary of prior approaches employed by researchers and their results; this includes the datasets, algorithms, classifiers, and techniques employed. This was helpful in formulating the method outlined in the next chapter.

Chapter 3

Research Methodology

3.1 Introduction

This chapter is dedicated to providing a detailed description of the proposed method. The datasets used, the methods employed, the data extraction methods, the keypoints dataset creation, training, the data analysis approach, and the hardware used are all included.

3.2 Methodology Pipeline

The methodology pipeline is seen in Figure 3.1. As depicted, first an appropriate Violent Video Dataset was downloaded to be used in the prototype. Afterward, the data to be used for training classifiers was extracted by making use of OpenPose Pose Estimation[1], YOLO Object Detection[2] and OpenCV Tracking API[3]. The data was labelled and exported to a CSV File. The CSV file was split into 80% training and 20% testing. A CNN with the required architecture was proposed and trained alongside an SVM with a linear

[1]OpenPose Pose Estimation: https://github.com/CMU-Perceptual-Computing-Lab/openpose
[2]You Only Look Once (YOLO): https://github.com/AlexeyAB/darknet
[3]OpenCV Tracking API: https://docs.opencv.org/4.x/dc/d6b/group__tracking__legacy.html

kernel with the data in the CSV file. Finally the performance of the resultant models was evaluated on the testing set.

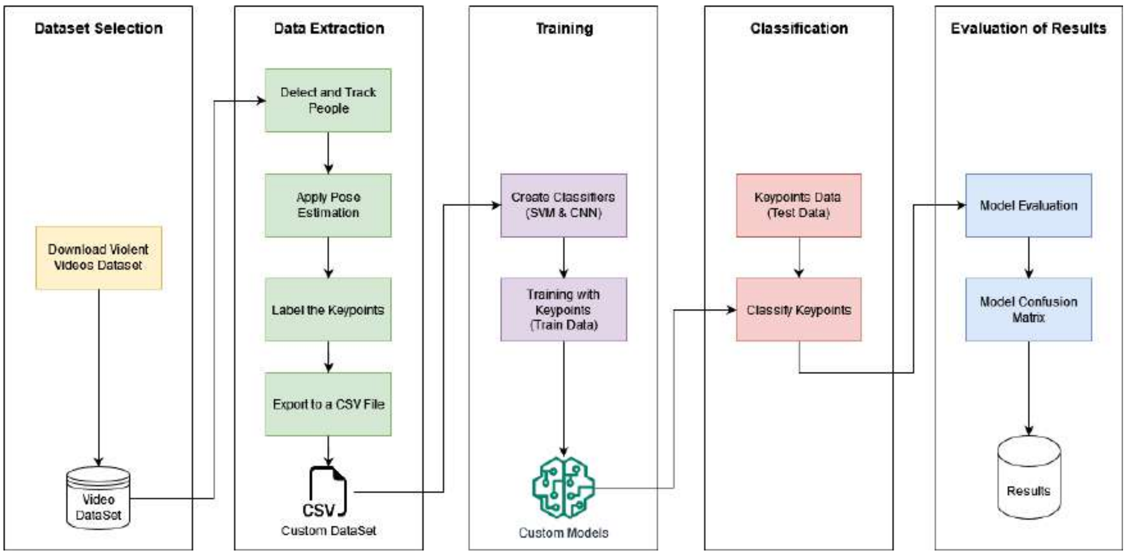

Figure 3.1: Research Methodology Pipeline.

3.3 Pose Estimation

As discussed in Sections 2.8.1 and 2.9.3, skeleton representation was utilised in multiple studies to classify the actions of individuals, as this has shown promising results when compared to other approaches. This approach was used in this research since when models are trained using the keypoints extracted, noise in images or the background are not considered. Therefore, when classifying the focus will only be put on the actions of the people unlike other approaches. Furthermore, this method of data extraction can be carried out using a simple RGB video with a Pose Estimation Algorithm as was done in the study conducted by Gatt et al. (2019). This is necessary if a dataset of standard camera's videos is to be used and the implementation is to be more reproducible in a real-life scenario.

From the algorithms outlined in Section 2.8, OpenPose (Cao et al., 2017) was deemed to be the most appropriate due to Part Affinity Fields (PAF). As discussed in Section 2.8, the number of humans detected in the image will have no effect on the overall performance of the system. This is important because in a violent activity, at least two people, and sometimes up to five, are involved. Furthermore, real-time performance is required since early detection of aggressive behaviour is critical in the detection of violence. Finally, as stated by Cao et al. (2017), because PAF collects data on the position and orientation of the human limbs, the algorithm is also more likely to predict accurately in general and in scenarios where postures overlap; which is a rather common occurrence during a fight.

3.4 Dataset Selection

As seen in Figure 3.1, selecting and downloading an adequate dataset was the first step of the pipeline. However, the quantity of violent and non-violent video clip datasets is limited. The ones listed in Section 2.10 are the most appropriate for this study, with Bianculli et al. (2020) 's proposal being the most promising. The other mentioned datasets are composed of clips with low resolution which hinders the Pose Estimation algorithm from extracting accurate keypoints. Furthermore, as mentioned by the authors, the proposed dataset attempts to prevent false positives by including varied rapid non-violent movement.

Unlike the other datasets, often only 2 to 6 participants are observed performing in a clip from this dataset, which is highly beneficial in preventing the Pose Estimation algorithm from detecting conflicting postures. The dataset is also split into two cameras, each of which records a clip from a different perspective, preventing the model from being able to only classify accurately from a particular angle. Lastly, since the clips are typically a short duration of 5 seconds and contains only a single action, trimming the clips is unnecessary and labelling is much more efficient.

3.5 Data Extraction

The pose estimation algorithm was employed onto the videos to calculate the skeleton representation of humans during the data extraction phase as seen in Figure 3.1. Temporal data is vital for action recognition, as was in works by Vyas et al. (2019) and Gatt et al. (2019). Hence, the model proposed in this study is fed a sequence of data extracted from frames.

3.5.1 Tracking

According to Nova et al. (2018), OpenPose is incapable of tracking people, making it difficult to keep a record of a single person's skeleton data across consecutive frames. The authors proposed using tracking algorithms provided by the OpenCV Tracking API as a solution.

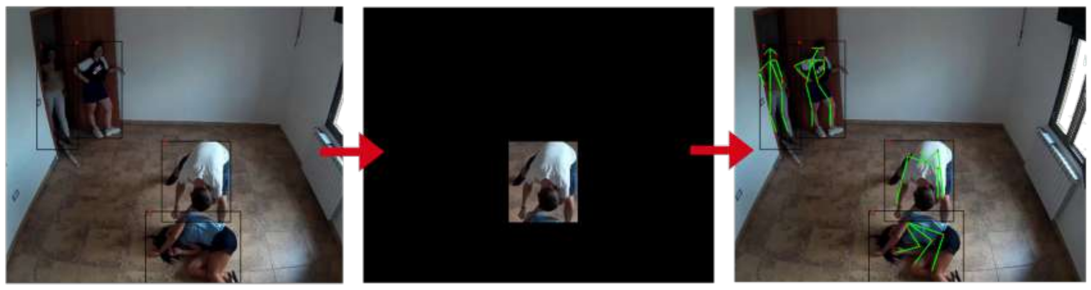

Figure 3.2: The Pipeline of generating the Bounding Boxes using YOLO, tracking individuals, separating them for clarity and finally Pose Estimation applied to extract skeleton data.

As a result of the authors' efforts, the tracker code-named CSRT from the same API was adopted for this research. The tracker is given a bounding box as the region of interest, which was produced automatically using Redmon et al. (2016) 's You Only Look Once (YOLO) Object Detection method. This technique was used to detect people in the image and then feed the Tracker API the extracted bounding boxes as seen in Figure 3.2. Each box was assigned a number that would eventually be used to relate to the individuals and to facilitate the labelling process.

The YOLO algorithm was also used to give additional information to the tracker overtime. As a result, the chances of limbs being left out of the image is reduced. This is further discussed in Section 4.2.1.

Due to the confidence threshold applied to OpenPose, it is likely for poses to be detached as some keypoints become missing. Therefore, once the trackers are established, the individuals are separated, as seen in Figure 3.2. This allows the detected keypoints to be safely attributed to a single individual, as well as preventing cluttering of poses. The extracted skeleton data could then be linked to a tracker, resulting in a record of an individual's posture across multiple frames.

3.5.2 Discarding of Data

As was outlined in Section 2.8, Pose Estimation is prone to collision of postures when individuals collide or overlap. This is a common occurrence in violent activity, albeit some measures were taken to prevent this, it is a hard challenge to overcome. In extreme cases where distorted clutter of keypoints was prominent in the video clip, the extracted data was discarded as it would only hinder the model during training.

Moreover, in the clips of the violent dataset utilised, it is common for a bystander to stand still in a pose for the duration of the clip. Therefore, the chances are that the samples extracted would be identical for that individual. Any duplicate data was discarded and only a single copy was maintained for training. This is essential to prevent the model from being fed the same data more than it should be which eventually results in overfitting.

3.5.3 Missing Keypoints

The application of OpenPose in this research had a confidence threshold applied, so that any joint predicted below said threshold would be discarded as it is prone to inaccuracy. However, this creates an issue in which a keypoint may be missing, preventing the data from being fed to the model and resulting in no classification. Hence, the value of the missing keypoints was filled in with that of (0, 0). According to Xu et al. (2018), filling the data with a computed keypoint, as was done in the research conducted by Vyas et al. (2019), is the incorrect approach as this may disturb the normal patterns of the action and potentially downgrades the performance of the model. According to Xu et al. (2018), it is preferable to simply fill in the missing points with zeros, which is similar to the approach of Gatt et al. (2019) and Yadav et al. (2020). To conclude whether this is the ideal solution for the proposed approach, a test involving eliminating the threshold of OpenPose was conducted. This allowed low confidence keypoints within the data but results in having no missing keypoints. This test is further discussed in Section 4.6.2.

3.5.4 Exporting to a CSV File (Keypoints Dataset)

Since the clips are short and generally only composed of a single action per person, it is safe to label the poses extracted for an individual as ‘Violent’ or ‘Non-Violent’. The Tkinter[4] library was used for this task and a simple GUI was built for labelling the data per video. The UI is depicted in Figure 3.3, where ’Violent’ actions can be marked by ticking the checkbox, while those left unticked are labelled as ’Non-Violent.’

The data was then normalised as was done in studies conducted by Gatt et al. (2019), Nar et al. (2016) and Xie et al. (2016), as they have discussed that this aids the model in the training procedure to efficiently process the data.

[4]Tkinter Library: https://docs.python.org/3/library/tkinter.html

Finally, the keypoints data from three successive frames were combined into a single sample, resulting in a time-series sample that preserved the video's sequence. This was done for each person in the video, which was then labelled appropriately and exported to a CSV file using the Pandas[5] library. Alongside each keypoint, OpenPose outputs the confidence score of each joint. This data was also exported into a separate CSV file to be used in tests described in Section 4.6.1.

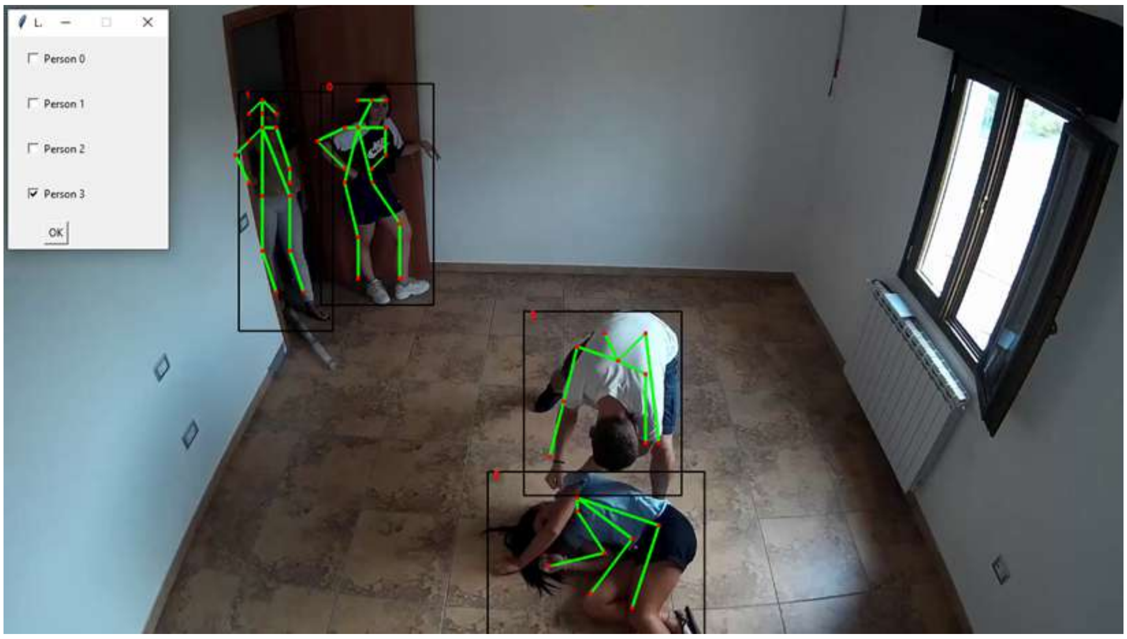

Figure 3.3: The labelling process with Tkinter GUI.

3.6 Model Training

The proposed dataset of keypoints comprised of 18,783 samples. They were all imported using Pandas and then augmented to fit a suitable proposed model. The data is used in the fitting process and for testing the performance of the resulted models.

[5]Pandas Library: https://pandas.pydata.org/

3.6.1 Data Processing

OpenPose alongside each keypoint, returns the respective confidence score. The confidence data was imported with the samples from a separate CSV file, and a mean value was determined for each sample. If the mean confidence score is below the given threshold, the sample would be ignored. This technique ensures the accuracy of the samples used for training the model. Tests described in Section 4.6.1, were performed to determine the ideal threshold value.

Due to the order of how the samples were stored, it is likely that a sample is followed by another sample with the same label. Therefore, to increase the randomness of the data being fed to the model, the samples are shuffled. This is done to ensure that when the data is split into the train and test sets, the data per label ratios are similar in both sets. This could also aid in preventing overfit models. The shuffling made use of a seed for reproducibility.

The numpy array containing the samples was reshaped to tensor shape of (S, 3, 38). S represents the total number of samples after some of the data is discarded and further reduced by the mean confidence threshold. The second number refers to the 3 time-steps formulating the time-sequence data. Finally, the 38 corresponds to the number of keypoints extracted by OpenPose, where each of the 19 keypoints has two values, the x and y axis. This data format was inspired by the work done by Gatt et al. (2019), where the authors have also used similar data that was passed to a CNN for classifying temporal information. The samples were then split into 80% training and 20% testing, following the split ratio of Vyas et al. (2019).

3.6.2 CNN Model Architecture

The proposed CNN model's architecture is illustrated in Figure 3.4. It consists of an input layer with a tensor shape of (S, 3, 38) as mentioned earlier. The input is passed to the hidden layers, starting with two Conv1D layers for feature extraction from the data passed. The Conv1D layer is used due to its potency with processing time-sequence data as mentioned and practiced in studies conducted by Basulaiman & Barati (2021), Kiranyaz et al. (2015) and Yang et al. (2020).

A dropout layer with a rate of 0.1 is then utilised to randomly remove a fraction of the data. These layers are used throughout the model to improve the generalization and reduce the chance of an over-fit model as mentioned by Albawi et al. (2017), Dwivedi et al. (2019), Mu et al. (2016), Narejo et al. (2021), and Sultani et al. (2018). Tests were conducted described in Section 4.7 to identify the ideal dropout rate.

As seen in figure 3.4, the features extracted by the Conv1D layers are fed into a Long Short-Term Memory (LSTM) layer. They are used by this layer to learn data relationships between time-steps. Combining both CNN and LSTM is a technique used by Yang et al. (2020) to predict time-sequence data, but also by Gatt et al. (2019) for action recognition.

The features are subsequently down sampled using a Max Pooling1D layer, which is used to reduce their spatial size. Moreover, the features are further compressed to a 1-dimensional array by a Flatten Layer which prepares the data for the next layers. Some of the data is discarded in another subsequent Dropout layer with a rate of 0.1.

The data is then fed to a Dense layer to compress the data, a Dropout layer of rate 0.3 and finally to another Dense layer which compresses the data into a single neuron. This is where the classification is outputted, and the input is determined to be either 'Violent' or 'Non-Violent'. Inspired by the works of Xu et al. (2018), an Adam optimizer was

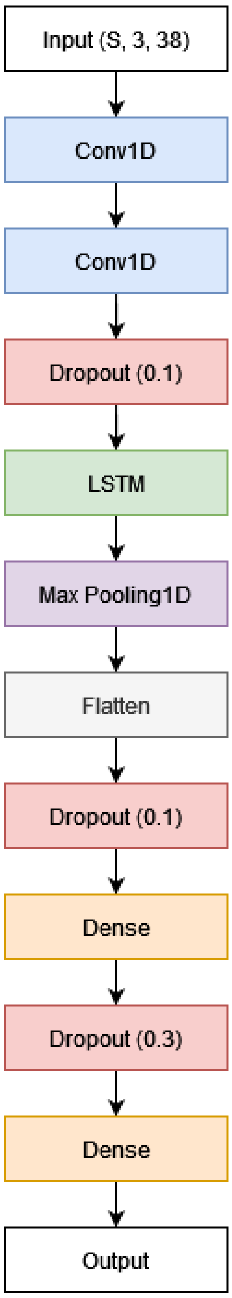

Figure 3.4: Proposed CNN model architecture.

used along with the binary cross-entropy loss function as the model is expected to make a binary classification.

3.6.3 Fitting Process

Python3 was used to utilise Tensorflow through the Keras interface to build, compile and fit the CNN model. It was trained in a supervised method for 50 epochs with a batch size of 32. The patience callback was utilised and was set to a value of 5. Such practice is common in various studies (Basulaiman & Barati, 2021; Gatt et al., 2019; Vyas et al., 2019; Xu et al., 2018) as it allows for early stopping when the model stops improving after a given number of epochs. This is used to prevent the model from overfitting.

An SVM model was also trained to compare the performance to the proposed CNN. The SVM model was built using a Linear kernel and fit with the SciKit-Learn library using Python3 as well. However, the shape of the data was changed due to the SVM limitations when it comes to the data dimensions in the input. Therefore, the data was reshaped to (S, 114), where S represents the number of samples and 114 are the number of features.

The training was done on a system with a 3.40GHz Intel Core i7-6700 CPU and a 6GB Nvidia Geforce GTX1060 graphics card.

3.7 Model Results

The results were based on the quantitative data derived from the model's post-training report. Both Keras and SciKit-Learn output data such as Accuracy, Precision and F1-Score which were analysed to determine and compare the performance of each classifier.

3.8 Conclusion

In this chapter, the process employed was described, alongside the reasoning behind the decisions taken. In the following chapter, the effects of the techniques implemented is described along with the analysis and description of the results achieved.

Chapter 4

Analysis of Results and Discussion

4.1 Introduction

This chapter will focus on discussing the proposed methodology such as the application of the algorithm, the extraction of the data, the classifiers' results, the training undertaken, and discussion of the accuracy achieved.

4.2 Detect and Track People

To extract time-sequence posture data of an individual, first the people within each clip were detected using the You Only Look Once (YOLO) Object Detection algorithm and then tracked using the OpenCV Tracking API.

4.2.1 YOLO Object Detection

First the individuals within a clip had to be detected, therefore the YOLO algorithm was employed on the dataset proposed by Bianculli et al. (2020). As seen in Figure 4.1, the YOLO algorithm was capable of detecting very accurately the individuals in the clips

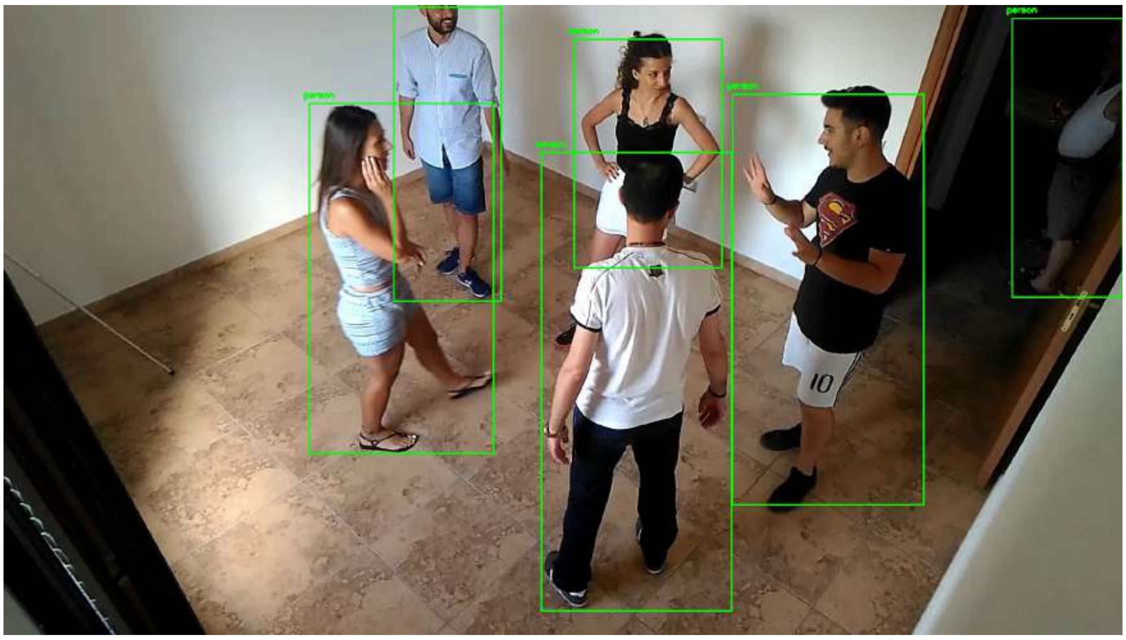

Figure 4.1: YOLO algorithm on the dataset proposed by Bianculli et al. (2020).

of the utilised dataset; even ones that are overlapping. This was very important for the tracking process as the bounding boxes would be utilised for the tracking sequence. The accuracy is most likely due to the high resolution of the dataset and the rather small amount of people presented in the frame.

In uncommon scenarios, the YOLO algorithm would incorrectly predict the people in the frame as seen in Figure 4.2 and Figure 4.3. Due to the overlap in hugging in Figure 4.2, the algorithm detected only a single person. In such cases, the data was discarded as there would only be a single tracker for two individuals, therefore the skeleton data would be inaccurate. However, this error only occurs on major overlapping, as can be seen on the previous frames separate bounding boxes were predicted and hence could have been used for tracking.

On the other hand, as seen in Figure 4.3, in a single individual, the algorithm predicted two separate bounding boxes. In such cases, the extra bounding boxes would be treated as an addition to the nearest tracker, and so integrated and treated as one as seen in right side of Figure 4.3. This most likely happens due to the abnormal poses presented in violent activity.

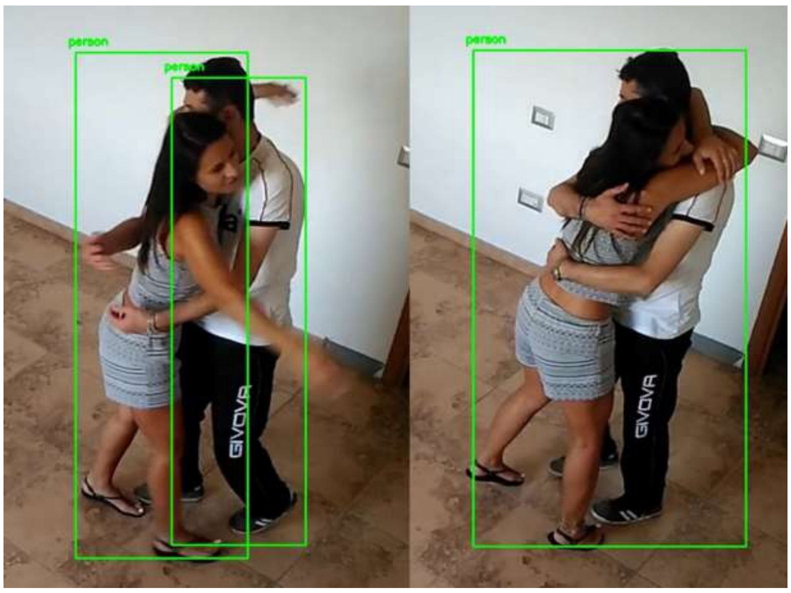

Figure 4.2: An example of YOLO algorithm predicting a single bounding box when two individuals collide.

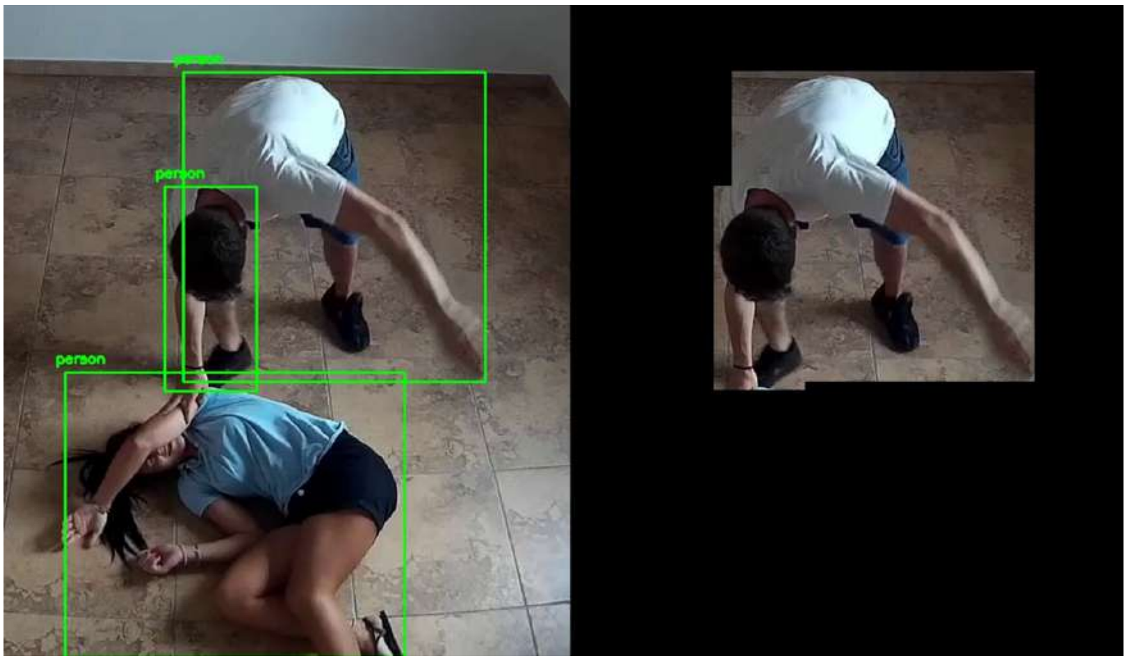

Figure 4.3: An example of YOLO algorithm predicting two bounding boxes for a single person (left). The bounding box being combined for better posture prediction (right).

4.2.2 OpenCV Tracking API

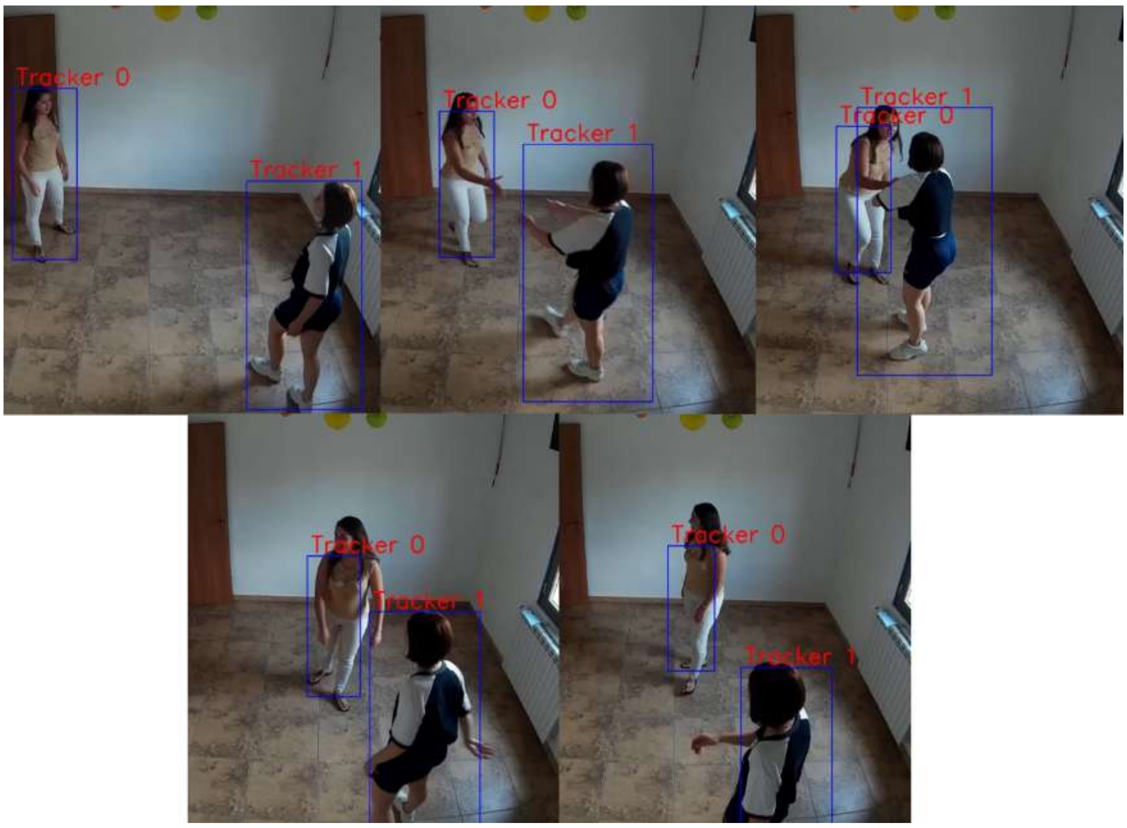

Figure 4.4: OpenCV Tracking API, tracking people and maintaining individuality.

As seen in Figure 4.4, using YOLO to feed the OpenCV Tracking API the bounding box was a useful procedure to track people and maintaining their individuality. This can be seen by the numbers allocated above each detected person. This consisted even after colliding with another tracker, making the tracker reliable.

That being said, there were cases where the tracker would lose focus as seen in Figure 4.5. This is caused by an overlap that occurs and the tracker would then follow other elements that were in the area instead. In such scenarios the data was discarded as it would only hinder the training process of the classifier.

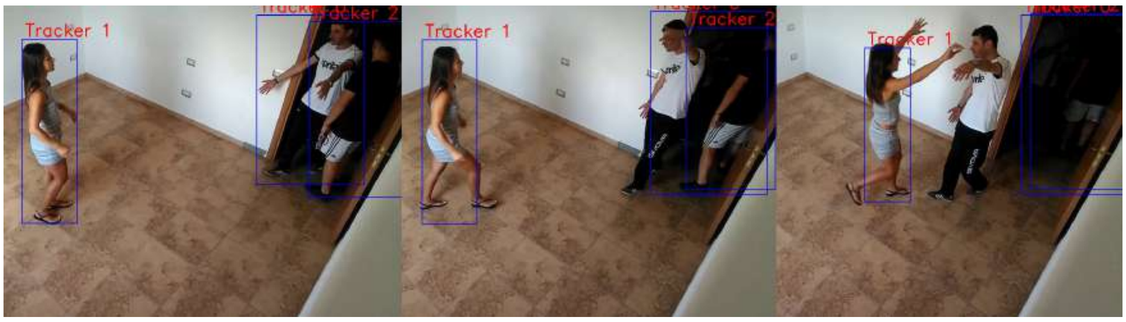

Figure 4.5: OpenCV Tracking API, losing focus on individuals that overlapped.

4.3 Pose Estimation

As the Pose Estimation was implemented within the prototype, a variety of techniques could have been utilised; each would produce a different result.

4.3.1 Single Person vs Multi Person OpenPose

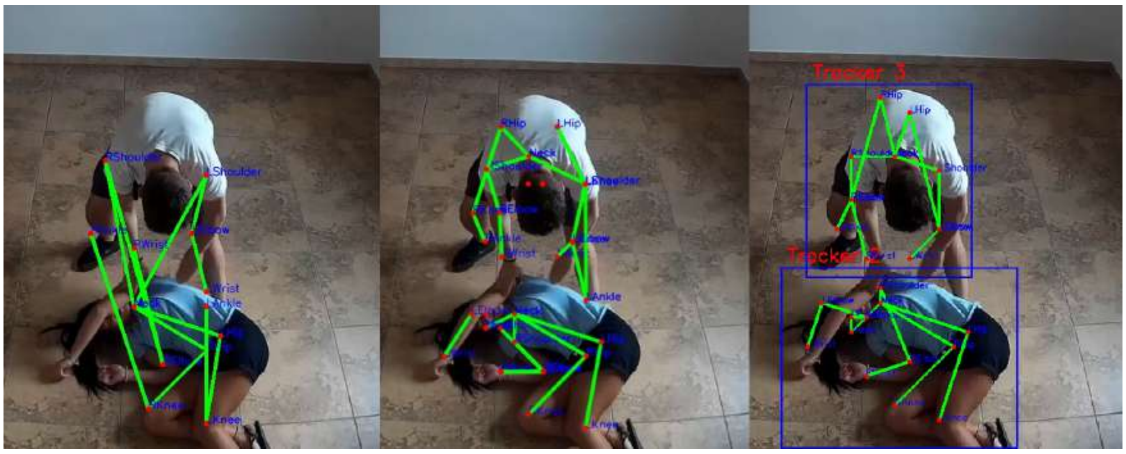

Figure 4.6: Single Person OpenPose vs Multi Person OpenPose vs Single Person Open-Pose with Trackers.

Since in violent activity, more than a single person is involved, Single Person OpenPose could have not been used as seen in Figure 4.6. The first image shows how Single Person OpenPose blended the poses of every person involved together, and the extracted skeleton representation would be inaccurate.

Even though Multi-Person OpenPose would seem to be the ideal solution, this still introduced other challenges. As seen in the second picture of Figure 4.6, some poses were being divided into smaller poses. The keypoint 'Left Hip' of the violent person and the 'Left Elbow' of the person on the floor are separated from the rest of the body. In this case, this would result in two separate poses for each person. This is most likely due to the confidence threshold applied to OpenPose. As a result, some joints are missing, leaving the 'Left Hip' without keypoints to connect to. This occurs when the threshold is set to 0.1; however, setting it to 0 would create inaccurate and unnecessary keypoints which may hinder the model's training process.

The proposed solution to this is to apply Single Person OpenPose on the trackers. As seen in the third image of Figure 4.6, this will separate each individual, and the keypoints found are declared to belong to that individual. This method ensures the accuracy of the estimated poses however increases the computational cost. As seen in Figure 4.7, the proposed approach has also proven to work when numerous individuals are present in the image.

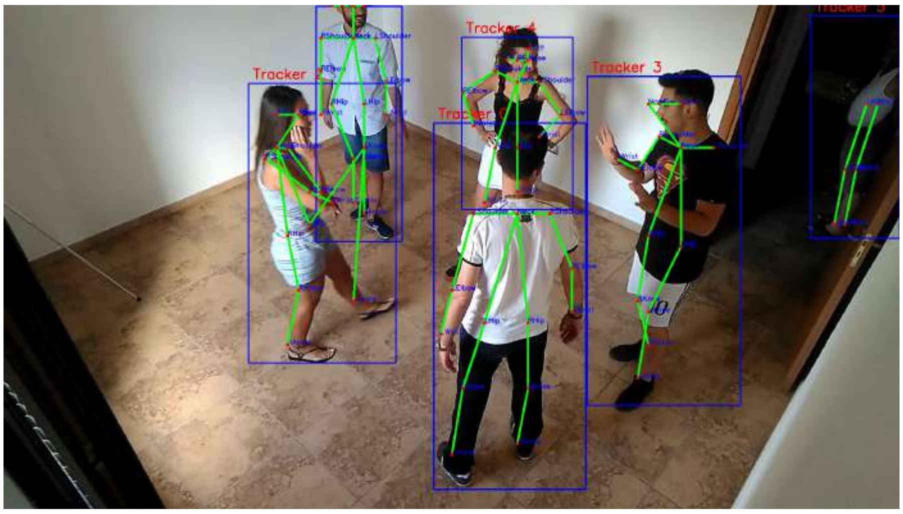

Figure 4.7: Single Person OpenPose with Tracker on multiple people in the image.

4.3.2 Collision of Poses

As discussed by Cao et al. (2017), even if Part Affinity Fields (PAF) are used, in some cases the poses still merge when major collision has occurred, or uncommon poses are present in the image. This is a fundamental problem to Pose Estimation that is challenging to solve, as some limbs can be hidden in the image when people overlap and the algorithm would have no reference to the missing limb. If the collision occurs for the vast duration of the clip, the data is discarded. However, in scenarios where the collision is minor and short-lasting, the data was retained.

4.4 CUDA and cuDNN

Hardware	**Time Taken (Average)**
CPU	8.2s
GPU	1.3s
GPU (2 Trackers)	0.8s

Table 4.1: Average Time Taken to process a single frame with CPU vs GPU(CUDA).

The CUDA version of both Tensorflow and OpenCV were utilised to significantly improve the runtime of the algorithms used. As seen in Table 4.1, the time taken to predict a single frame using the trackers and OpenPose was reduced to an average of 1.3s per frame by utilising the GPU instead of the CPU. The time taken differs with the amount of people within the frame due to the number of trackers used; for example, when there are only two people in the image, the time taken drops to an average of 0.8s. The CUDA version was essential for extracting data from Bianculli et al. (2020)'s dataset and classifying poses with the trained model in a shorter amount of time. Installing the Tensorflow version that enables GPU usage has also sped up the training process, which is further discussed in Section 4.8.

4.5 Labelling Actions and CSV File

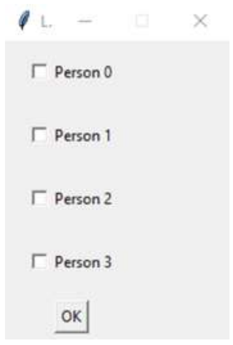

Figure 4.8: The proposed Tkinter UI used for labelling.

As discussed in Section 3.5.4, Tkinter was used for labelling the poses from the dataset proposed by Bianculli et al. (2020) and exporting the data to a CSV file. The UI created is shown in Figure 4.8. As described in Section 3.5.4, the poses of individuals would be labelled as 'Violent' if the checkbox of the corresponding tracker number is ticked.

The dataset consisted of 350 clips, and 18,782 samples were collected after extracting data from all of the videos and discarding any that would compromise the models' accuracy. There were 10,285 samples labelled as 'Non-Violent,' while the remaining 8497 were labelled as 'Violent'. Each sample comprised of the co-ordinates of the joints where each had their x and y values separate and normalised. Since OpenPose extracts 19 joints and 2 different axis, this resulted in 38 features for every frame. Each sample consisted of data from 3 sequential frames, which results in each sample having 114 features.

4.6 Optimising the Data

Before the data was fed to the classifier for training, further techniques were employed to ensure the quality of the data, and to optimise the results obtained.

4.6.1 Discarding Poses with Low Mean Confidence Score

When using OpenPose, each pose extracted from the clips would have the corresponding confidence score for each joint. Low confidence score indicates that the joint is more likely to be inaccurate, which could disrupt the typical movement patterns of the action. The confidence score was used to calculate the mean confidence of each sample and determine which were deemed to be more accurate than others.

During tests a threshold was applied, and samples that fell below that threshold were removed and not included in the fitting procedure. The aim of this test is to identify the ideal threshold and remove any data that could be considered as inaccurate and might hinder the performance of the classifier.

Mean Confidence	Data Samples	Accuracy
0	18,782	74%
0.1	18,116	75%
0.2	17,115	75%
0.3	15,904	78%
0.4	14,319	76%
0.5	11,837	69%
0.6	8130	62%
0.7	4433	59%
0.8	587	39%
0.9	0	N/A

Table 4.2: The Mean Confidence threshold applied and the resultant accuracy.

As seen in Table 4.2 as the mean confidence threshold is increased, the number of samples used for training decreases. From a threshold of 0 to 0.3 there was a steady increase in accuracy, and this is due to removing poses that are more prone to randomness and therefore a pattern of movement that does not resonate with the others. As the threshold number was further increased, from 0.4 to 0.8, the accuracy drastically decreases. The main reason being that the majority of the data has been discarded from the dataset, and the model lacked data for training.

Every posture has a reduced mean confidence score due to missing keypoints; these have a confidence value of 0. Therefore, a mean of 0.3 is sufficient to eliminate the severely inaccurate poses while maintaining 15,904 samples of data that is considered of adequate quality.

4.6.2 Missing Keypoints vs No Confidence Threshold

In the study conducted by Xu et al. (2018), the authors recommended to fill any missing keypoint with co-ordinates (0,0) instead. To verify whether this technique is the ideal approach in the proposed solution, a test has been conducted. This involves changing the confidence threshold of OpenPose. There were two scenarios: the first one involved applying the threshold and hence some keypoints were missing. Whereas the second one involved completely removing the threshold and all keypoints would be extracted, including low confidence ones.

Scenario	**Accuracy**
Threshold Applied	78%
No Threshold	72%

Table 4.3: Threshold Applied vs No Threshold and the resultant accuracy.

As can be seen from Table 4.3, the results align with what was discussed by Xu et al. (2018). The results has shown that the model achieved lower accuracy on the testing set when no threshold was applied to OpenPose. Therefore, this concludes that it is more ideal to replace the missing keypoints with a value of (0,0) rather than filling the keypoint with a possible inaccurate value. The reason behind this is most likely due to the keypoint not following the flow of the action being conducted. Thus, making it harder for the model to recognise patterns in the action.

4.7 Dropout Layer Rate

As discussed in Section 3.6.2, the proposed CNN model's architecture consists of three separate Dropout layers throughout to reduce the chance of an overfit result. As seen in Table 4.4 a Dropout Rate of 0.5 throughout the architecture has proven to produce the best results for the proposed methodology. This is most likely because any rate below 0.5 would result in a more overfit model as not enough data is in rotation for the training process. However, as the dropout rate increases beyond 0.5, the accuracy decreases. This is probably due to the model being unable to observe data repetitively and learn from it frequently enough in epochs; this is because a large amount of data is being discarded within these layers.

Total Dropout Rate	**Accuracy**
0.3	78%
0.4	78%
0.5	79%
0.6	76%
0.7	74%

Table 4.4: Total Dropout Rate in the CNN architecture and the resultant accuracy.

4.8 Training

A CNN and an SVM were trained to identify the ideal classifier in recognising violent behaviour.

4.8.1 CNN

First, to initiate the model fitting process with the proposed dataset, the CNN model architecture was determined and built. The dataset was split into 80% training and 20% testing and the model was set to train for 50 epochs. Due to early stopping, the model ended the process after 33 epochs. The patience callback was executed after the model stopped improving for 5 epochs, making the whole process last 2 minutes and 47 seconds. The short duration is most likely due to using the GPU version of TensorFlow, the model's architecture is simple and the data is not as vast as an image or a video. Moreover, despite the large number of samples, the data is extremely simple being numeric with a small shape. Figure 4.9 and Figure 4.10 show the accuracy attained and the loss after each epoch during the fitting process. The proposed CNN model achieved 79% accuracy on the testing set.

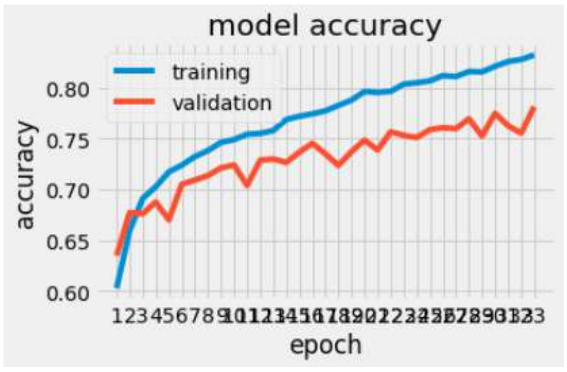

Figure 4.9: Model Accuracy Graph of the proposed CNN model.

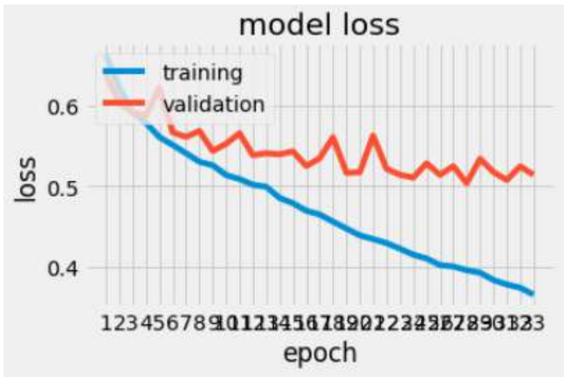

Figure 4.10: Model Loss Graph of the proposed CNN model.

4.8.2 SVM

For testing purposes an SVM classifier with a linear kernel was trained with the same dataset and split ratio between training and testing sets. As with the details of the proposed CNN outlined in Section 4.6.1, the mean confidence of the poses was likewise adjusted to 0.3. However, as mentioned in Section 3.6.3, the data was reshaped to (S, 114) which means that the time-based data was compressed.

The model was set to train for 50 epochs and the fitting procedure lasted 35 seconds. The short training duration is most likely due to the simplicity of the linear kernel architecture. The proposed SVM model achieved 63% accuracy on the testing set.

4.9 Results and Model Performance

From the predictions made on the testing set, the performance and the measurements of the trained models, were calculated and extracted.

4.9.1 CNN vs SVM

Looking at the accuracy achieved by both models, the CNN outperforms the SVM by a large margin, where the CNN achieved 79% and the SVM managed 63% accuracy. The main reason the difference is so drastic is due to 1DConv and LSTM layers used in the CNN architecture; these are potent in processing time-based data. Moreover, for the SVM, the data had to be reshaped due to limitations, which compressed the time-based data into a single dimension. This could have lost the time element which resulted in worse performance. As a result, the proposed CNN was determined to be better, and further testing was conducted on it.

4.9.2 CNN Measures

Further tests were done on the CNN model to extract related statistics when predicting video footage. As seen in Table 4.5, a Confusion Matrix is depicted showing the post-training results on the testing set. The test results shows that the model could predict

‘Non-Violent’ action better than ‘Violent’ activity. This could be because overall there were more data extracted of ‘Non-Violent’ behaviour and had more data to train with.

		Predicted	
		Non-Violent	Violent
Actual	Non-Violent	2742	562
	Violent	731	2024

Table 4.5: Confusion Matrix on Testing Set of the proposed CNN model.

Due to making use of the dataset proposed by Bianculli et al. (2020), which included ‘Non-Violent’ rapid-moving actions, the amount of false positives is less than the false negatives. However, there were still a fair number of false postives, this might not be an issue related to the dataset, but rather a problem that could have been solved if more frames were used per sample. This would result in more data per sample to predict from and allow the model to be able to observe the difference between a larger number of frames. This might address the issue since a ‘Non-Violent’ fast moving action can still be seen as ‘Violent’ in some scenarios. This occurs when only 3 frames from the action are taken as context for prediction and if the change in the keypoints’ co-ordinates is drastic enough.

	Precision	**Recall**	**F1-Score**
Non-Violent	79%	83%	81%
Violent	78%	73%	76%
		Final Accuracy	**79%**

Table 4.6: Metric Scores on Testing Set of the proposed CNN model.

Keras was used to calculate other measures as seen in Table 4.6. As discussed beforehand, the model’s final accuracy is 79%. However, looking at the other statistics, the Violent class has a Precision of 78%, Recall of 73% and F1-Score of 76%. This is most likely a case of overfitting. Observing the Model Loss Graph in Figure 4.10, the way the graph plateaus further underlines this. This might be due to the dataset size as there might have been lack of training data. Violent behaviour cannot be described with a single uniform pose, but there are a wide range of ways that one can act violently. Some unique poses might have been included in the testing set that the model has never experienced before which hindered the results.

Lastly, as previously stated, the model seems to predict Non-Violent activity more accurately with a Precision score of 79%, Recall of 83% and F1-Score of 81%. This is most likely due to there being more data related to ‘Non-Violent’ behaviour in the dataset.

4.9.3 Predicting Violent and Non-Violent Activity

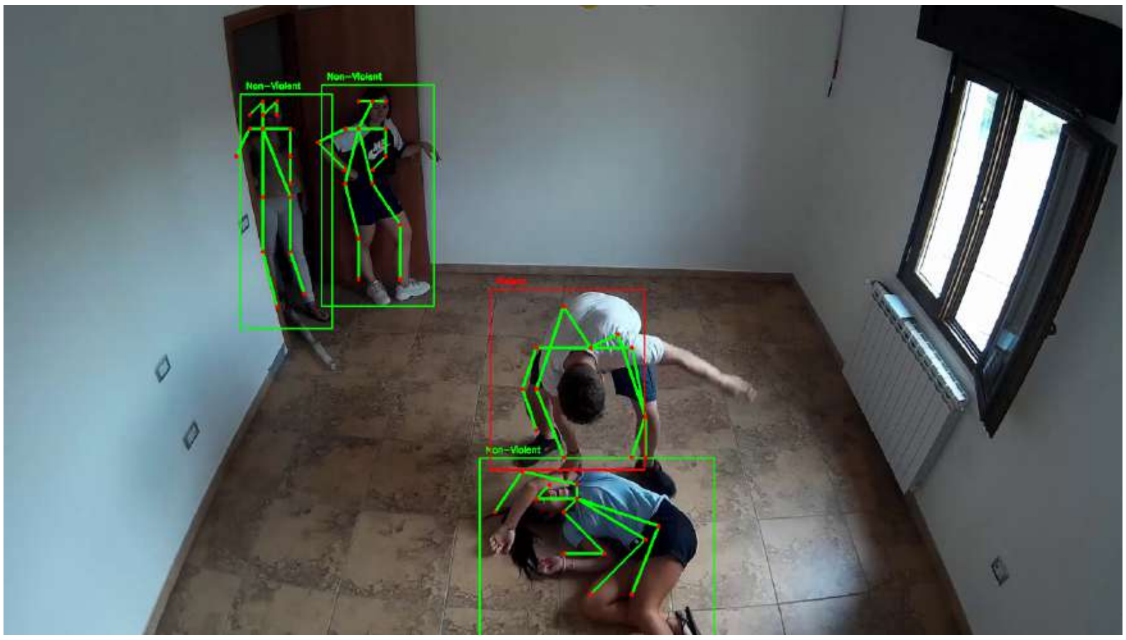

Figure 4.11: The proposed CNN model predicting violent and non-violent behaviour from a video.

The model could predict correctly as seen in Figure 4.11, where the person committing violent actions was labelled correctly whereas the bystanders and the victim are classified as ‘Non-Violent’. Any stationary individuals as seen in the top left of Figure 4.11 are most of the time predicted as ‘Non-Violent’, this is because the model predicts no movement which is a clear indication that the person is not acting violently. Another scenario is illustrated in Figure 4.12 where two individuals are acting violently towards

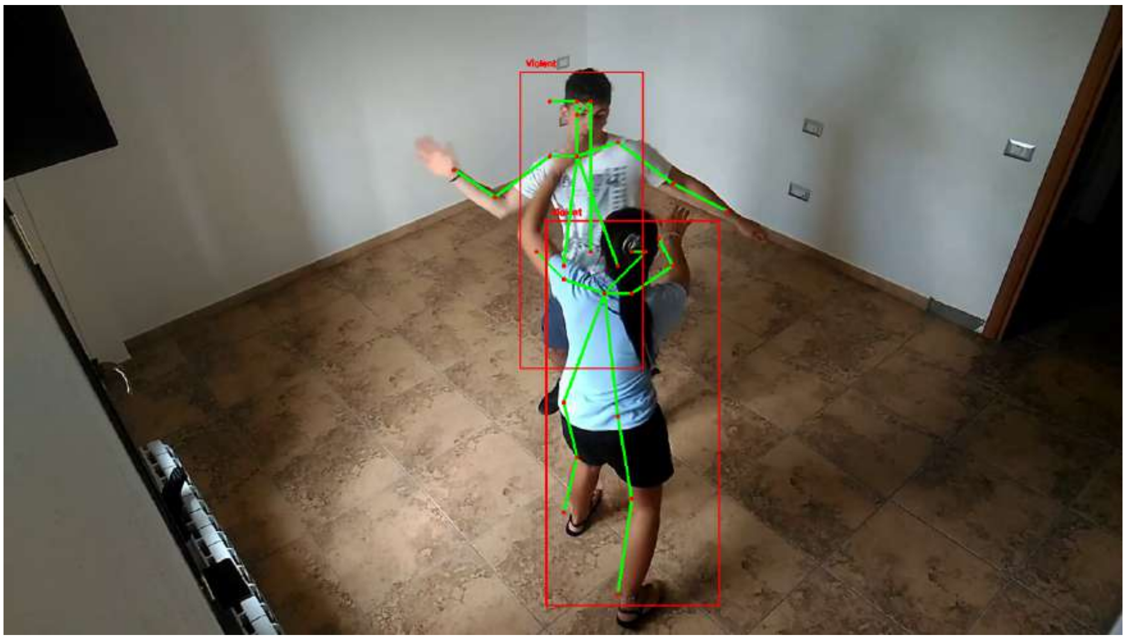

Figure 4.12: The proposed CNN model predicting two violent actions correctly.

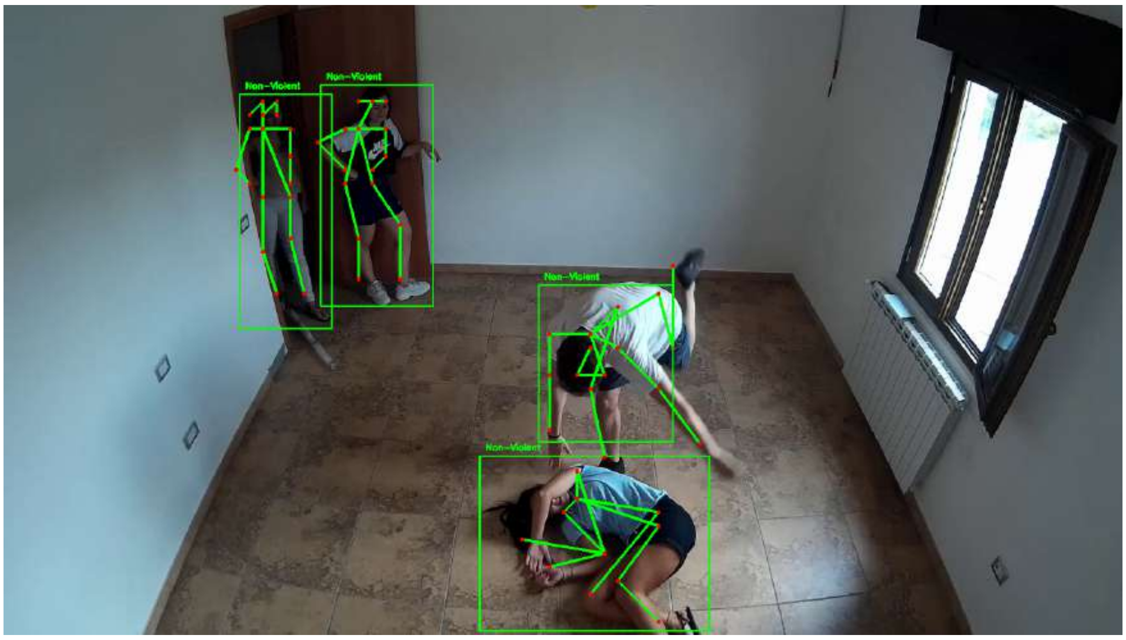

Figure 4.13: The proposed CNN model inaccurately predicting violent poses.

each other, and the proposed model was capable of predicting correctly.

That being said, the accuracy of the model is not perfect, therefore as seen in Figure 4.13, the model can still inaccurately predict poses even within the same clip. This could be that within the three frames that are being used for prediction, not enough data was presented for the model to conclude that a pose was in fact violent.

4.9.4 Non-Violent fast action

To further determine the performance of the proposed model, the model was used to predict non-violent fast actions. As seen in Figure 4.14, the model was used to predict a handshake and successfully classified both individuals involved as ‘Non-Violent’. This is most likely because the training process included non-violent fast actions to prevent false positives as discussed in Section 3.4 and in this clip there was no overlap and the poses extracted were very clean.

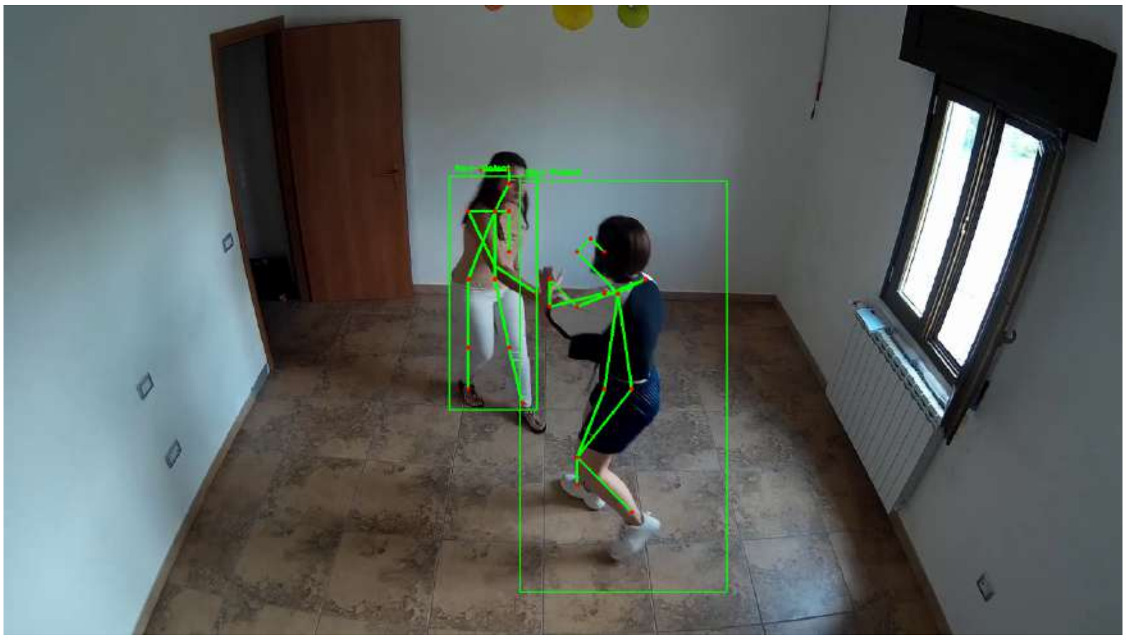
Figure 4.14: The proposed CNN model predicting a handshake as Non-Violent.

The clip referenced in Figure 4.14 involved 126 frames and in only two of them, seen in Figure 4.15, one of the individuals was classified as ‘Violent’. The poses in these frames are merged due to overlap which could have resulted in the incorrect classification.

Another case illustrated in Figure 4.16 involved two individuals approaching each other

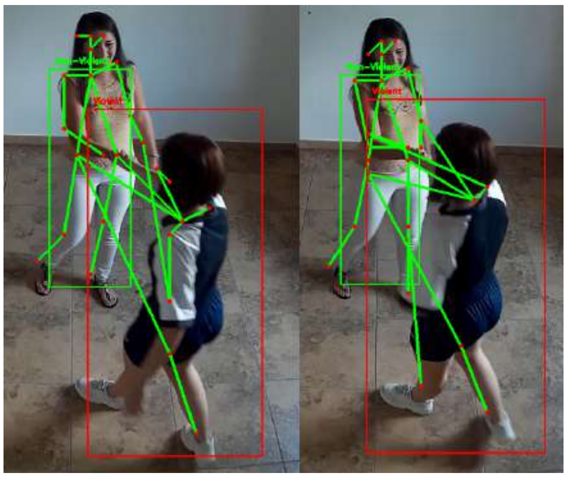

Figure 4.15: The proposed CNN model incorrectly predicting Violent actions due to overlap of two poses in two separate frames.

for a hug which the proposed CNN model classifies correctly as ‘Non-Violent’.

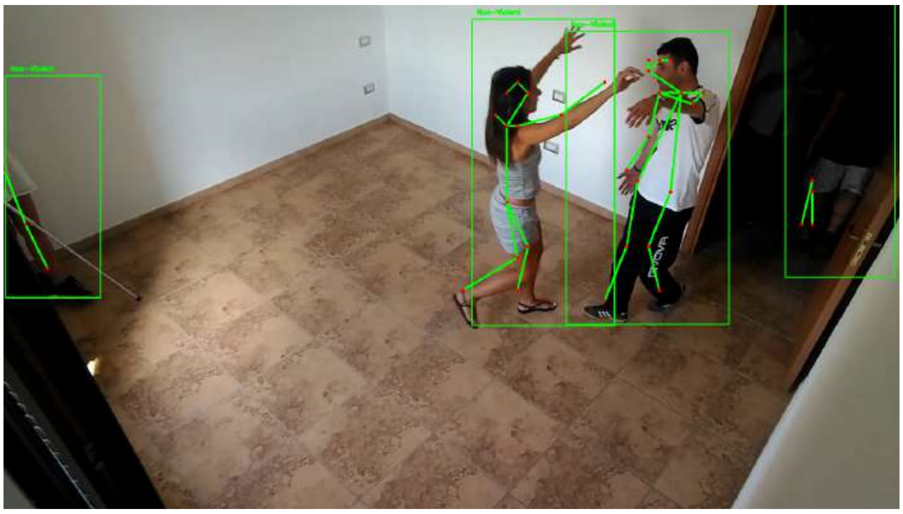

Figure 4.16: The proposed CNN model correctly predicting a hug as Non-Violent.

Although on collision, as seen in Figure 4.17, the poses overlap, and the classifier misclassifies the action as ‘Violent’. As stated beforehand this is most likely due to the poses overlapping and hence the data being fed to the classifier was inaccurate.

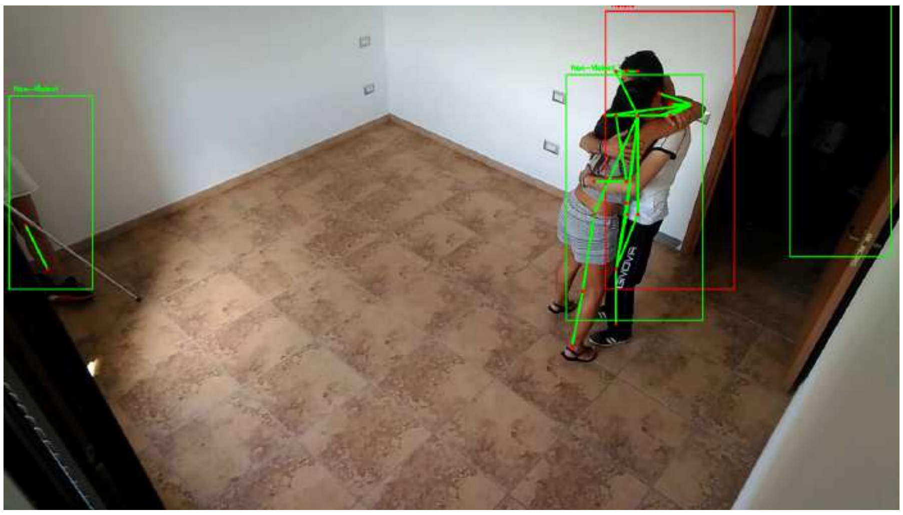

Figure 4.17: The proposed CNN model incorrectly predicting a hug as Violent.

4.10 Comparison with existing literature

Nova et al. (2018), proposed an SVM with a linear kernel to detect violent actions by using time-sequence data which achieved 89% accuracy. This could be due to utilising 3D data to fit the classifier, alongside engineered features and whether the individuals are touching. However, the authors discussed that they would have liked to have tested distinct video scenarios and improve the proposed solution to be capable of classifying more than two individuals. These are all challenges that were attempted to be solved in this work. In fact, the classifier in this research was actively tested with fast moving non-violent actions. Moreover, the proposed solution could classify any number of people, as it focuses on classifying each individual's posture independently and irrelevant of the surrounding. This is deemed to be important as violent behaviour can still be practiced within distance where two people are not touching. However, this comes at a computational cost and at the increased risk of posture collision if a crowd is present.

Gatt et al. (2019) came to a similar conclusion to this study, where pose estimation can be used to extract data from a normal camera and use said data to train a classifier. Moreover, the use of 1DConv layers alongside LSTM layers have proven in both studies that is an effective method in classifying time-sequence data. Albeit the actions being classified are different, the Precision achieved by Gatt et al. (2019) is higher than this study but rather comparable as well. However, in some scenarios the measurements where in fact much higher, but this could be difficult to trace the reason due to the different actions. One of the reasons could be that the authors retrieved data from 30 sequential frames, but in this study, only 3 were used to be able to classify as quickly as possible. Moreover, Gatt et al. (2019) opted for a Semi-Supervised method instead. The authors achieved real-time speeds, but the prototype made use of no trackers as the study focused on a single person within the frame. This is less performance heavy but can cause challenges when a multiple people are included in the scene.

The approach proposed by Vyas et al. (2019) achieved 72% accuracy in classifying atypical (ASD) behaviour. The actions being classified are different, therefore a direct comparison cannot be made, however the proposed model in this work has outperformed the model in the research conducted by Vyas et al. (2019). Similarly, Vyas et al. (2019) opted to train a CNN to classify actions however their approach included creating an RGB image code named ‘PoTion’ to represent the keypoints location overtime. This results in unable to make use of 1DConv layers in reading time-sequence data as was utilised in this study. The biggest setback could be that the RGB image representation contains data from 301 frames that translates to 10 seconds. The Data being fed to the CNN might have been too specific and thus the model might have been overfit. Moreover, it could only classify the poses after a 10 second delay which is too late for emergency calls, whereas the proposed method in this work, classifies after 3 frames which is a tenth of a second with a good performing system.

4.11 Limitations

Despite that Pose Estimation is a viable method for detecting violent behaviour, it is not a flawless solution. When people collide and overlap, or are found in unusual postures, the algorithm extracts inaccurate or a cluster of keypoints, making classification challenging. Furthermore, trackers tend to lose concentration on the person they are following in such situations. Another drawback was that when a crowd was exhibited in a clip, the algorithm tended to have major collision and thus incorrect predictions.

The data available was most likely the main constraint. Bianculli et al. (2020) proposed a dataset that was rich in data and had enough visual clarity to extract the information needed for this study with Pose Estimation. However, during the extraction of the data from the dataset, there were still issues with collisions and missing keypoints, resulting in less accurate values.

Albeit addressing the missing keypoints challenge as was recommended by Xu et al. (2018), this still hindered the predictions as less data is available for classification. Having all keypoints extracted, on the other hand, would be the ideal scenario because it would allow for improved model training and prediction.

Finally, despite making every effort to keep the pipeline light on resources, the prototype was still unable to work in real time. This could be owing to the high resolution necessary for the Pose Estimation algorithm to derive an accurate skeleton or needing trackers for every individual to keep a record of their posture. This could also be due to the system's hardware being slightly out of date. It is also worth noting that there could be inefficient component consumption throughout the pipeline.

Chapter 5

Conclusions and Recommendations

5.1 Evaluation of the Hypothesis

The hypothesis of this research was: by making use of Machine Learning, it is possible to train a classifier to detect violent behaviour with time-sequence pose estimation data collected from a standard camera's video clip. The objectives were to investigate what sort of dataset was needed, what kind of algorithm and framework could be employed, and the accuracy of the proposed solution in predicting violent behaviour. The proposed solution has shown, through the outlined results, that it is possible to distinguish between non-violent and violent behaviour through a sequence of posture data. Moreover the detection can be made using data extracted from a video footage of a simple camera without the need of additional equipment.

CNNs with 1DConv and LSTM layers have shown to be effective at classifying time-sequence data, however a sufficient number of frames must be utilised for optimal results. It was also concluded that eliminating posture with low mean confidence contributes to an overall better performing model. The proposed classifiers' measures has shown that the CNN outperformed the SVM. The proposed dataset in conjunction with the overall framework managed to achieve 79% Accuracy. Higher accuracy, on the other hand, is thought to be possible with a larger dataset free of colliding postures and further optimisation of the data.

5.2 Recommendations

Due to the general performance concerns and the need for the system to perform in realtime, lightweight solutions to the algorithm used: OpenPose, YOLO and the OpenCV Tracking API should be considered.

In this study, a major improvement that can be recommended is the development of a custom dataset from skeleton data generated in a controlled virtual 3D environment such as Unity[1]. The use of Motion Capture could also be considered for the creation of animation. This will eliminate the challenge of pose collision and overlap prior to model training. Such data would then be converted to 2D data and exported in a CSV file as was done in this research so that classification from a normal camera could still be possible. However, other solutions must be considered to eliminate the same problem when it comes to inference in realtime through a normal camera.

There are some changes that can be explored when it comes to the data being fed to the classifiers. Along with the pose keypoints, other researchers such as Nova et al. (2018), explored the method of engineering features to employ for training and classification such as velocity and limb angles. It is also recommended to consider a case in which the number of consequential frame data, which was three in this study, is increased. However, caution must be taken to prevent drastic increase in the frame count. Since, as the number of frames increases, the more specific the data becomes which can result in

[1]Unity: https://unity.com/

overfitting as seen in the study by Vyas et al. (2019).

When it comes to the classifier themselves, testing other types of classifiers, such as Random Forest (RF) or a non-linear SVM, could lead to interesting results. These techniques are worth investigating to be implemented in future studies and observed for their effect on the results of the prototype.

5.3 Future Work

The posture of a person could determine their behaviour and is essential in unarmed combat, however the detection of dangerous armed individuals can be done much earlier. The proposed solution can be enhanced by utilising Object Detection technology to detect dangerous objects such as a gun or a knife, as was done in the research conducted by Dwivedi et al. (2019), which is a clear sign someone might act hostile. However distinction must be made between people that are meant to have these objects such as a police officer or a chef.

Another recommended improvement is to introduce Facial Recognition. This enables the system to recognise a person for identification purposes if it suspects them of being hostile.

It is worth considering that some actions may take place outside of the camera's range of view but close enough to provide audio data. Mu et al. (2016) claims that their auditory approach in detecting violent scenes has shown promising capabilities, implying that such a system is worth exploring and could be implemented.

Finally, real-time training for continuous improvement by consistently feeding the classifier live data from a public environment should be considered. This would improve the overall system by significantly increasing the number of samples in the dataset.

Appendices

Project Code

1. Python Code for Data Extraction, Training CNN and SVM and Prediction:

- https://github.com/SeanFarrugiaMSD/Thesis

Bibliography

Albawi, S., Mohammed, T. A. & Al-Zawi, S. (2017), Understanding of a convolutional neural network, *in* '2017 international conference on engineering and technology (ICET)', Ieee, pp. 1–6.

Arceda, V. M., Fabián, K. F., Laura, P. L., Tito, J. R. & Cáceres, J. G. (2016), 'Fast face detection in violent video scenes', *Electronic Notes in Theoretical Computer Science* **329**, 5–26.

Basulaiman, K. & Barati, M. (2021), Sequence-to-sequence forecasting-aided state estimation for power systems, *in* '2021 IEEE Texas Power and Energy Conference (TPEC)', IEEE, pp. 1–6.

Bermejo Nievas, E., Deniz Suarez, O., Bueno García, G. & Sukthankar, R. (2011), Violence detection in video using computer vision techniques, *in* 'International conference on Computer analysis of images and patterns', Springer, pp. 332–339.

Bianculli, M., Falcionelli, N., Sernani, P., Tomassini, S., Contardo, P., Lombardi, M. & Dragoni, A. F. (2020), 'A dataset for automatic violence detection in videos', *Data in brief* **33**, 106587.

Calzavara, I. (2020), 'Human pose augmentation for facilitating violence detection in videos: a combination of the deep learning methods densepose and vionet'.

Cao, Z., Simon, T., Wei, S.-E. & Sheikh, Y. (2017), Realtime multi-person 2d pose estimation using part affinity fields, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 7291–7299.

Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L. & Zdeborová, L. (2019), 'Machine learning and the physical sciences', *Reviews of Modern Physics* **91**(4), 045002.

Cheng, M., Cai, K. & Li, M. (2020), Rwf-2000: An open large scale video database for violence detection, *in* '2020 25th International Conference on Pattern Recognition (ICPR)', pp. 4183–4190.

Dwivedi, N., Singh, D. K. & Kushwaha, D. S. (2019), Weapon classification using deep convolutional neural network, *in* '2019 IEEE Conference on Information and Communication Technology', IEEE, pp. 1–5.

Fu, E. Y., Leong, H. V., Ngai, G. & Chan, S. C. (2017), 'Automatic fight detection in surveillance videos', *International Journal of Pervasive Computing and Communications* .

Gatt, T., Seychell, D. & Dingli, A. (2019), Detecting human abnormal behaviour through a video generated model, *in* '2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)', IEEE, pp. 264–270.

Gracia, I. S., Suarez, O. D., Garcia, G. B. & Kim, T.-K. (2015), 'Fast fight detection', *PloS one* **10**(4).

Hassner, T., Itcher, Y. & Kliper-Gross, O. (2012), Violent flows: Real-time detection of violent crowd behavior, *in* '2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops', IEEE, pp. 1–6.

Jordan, M. I. & Mitchell, T. M. (2015), 'Machine learning: Trends, perspectives, and prospects', *Science* **349**(6245), 255–260.

Kiranyaz, S., Ince, T. & Gabbouj, M. (2015), 'Real-time patient-specific ecg classification by 1-d convolutional neural networks', *IEEE Transactions on Biomedical Engineering* **63**(3), 664–675.

Mu, G., Cao, H. & Jin, Q. (2016), Violent scene detection using convolutional neural networks and deep audio features, *in* 'Chinese Conference on Pattern Recognition', Springer, pp. 451–463.

Munea, T. L., Jembre, Y. Z., Weldegebriel, H. T., Chen, L., Huang, C. & Yang, C. (2020), 'The progress of human pose estimation: a survey and taxonomy of models applied in 2d human pose estimation', *IEEE Access* **8**, 133330–133348.

Nar, R., Singal, A. & Kumar, P. (2016), Abnormal activity detection for bank atm surveillance, *in* '2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)', IEEE, pp. 2042–2046.

Narejo, S., Pandey, B., Rodriguez, C., Anjum, M. R. et al. (2021), 'Weapon detection using yolo v3 for smart surveillance system', *Mathematical Problems in Engineering* **2021**.

Noble, W. S. (2006), 'What is a support vector machine?', *Nature biotechnology* **24**(12), 1565–1567.

Nova, D., Ferreira, A. & Cortez, P. (2018), A machine learning approach to detect violent behaviour from video, *in* 'International Conference on Intelligent Technologies for Interactive Entertainment', Springer, pp. 85–94.

O'Shea, K. & Nash, R. (2015), 'An introduction to convolutional neural networks', *arXiv preprint arXiv:1511.08458* .

Papandreou, G., Zhu, T., Kanazawa, N., Toshev, A., Tompson, J., Bregler, C. & Murphy, K. (2017), Towards accurate multi-person pose estimation in the wild, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4903–4911.

Pisner, D. A. & Schnyer, D. M. (2020), Support vector machine, *in* 'Machine learning', Elsevier, pp. 101–121.

Ramzan, M., Abid, A., Khan, H. U., Awan, S. M., Ismail, A., Ahmed, M., Ilyas, M. & Mahmood, A. (2019), 'A review on state-of-the-art violence detection techniques', *IEEE Access* **7**, 107560–107575.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016), You only look once: Unified, real-time object detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 779–788.

Serrano, I., Deniz, O., Espinosa-Aranda, J. L. & Bueno, G. (2018), 'Fight recognition in video using hough forests and 2d convolutional neural network', *IEEE Transactions on Image Processing* **27**(10), 4787–4797.

Sultani, W., Chen, C. & Shah, M. (2018), Real-world anomaly detection in surveillance videos, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Szeliski, R. (2010), *Computer vision: algorithms and applications*, Springer Science & Business Media.

Vyas, K., Ma, R., Rezaei, B., Liu, S., Neubauer, M., Ploetz, T., Oberleitner, R. & Ostadabbas, S. (2019), Recognition of atypical behavior in autism diagnosis from video using pose estimation over time, *in* '2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)', IEEE, pp. 1–6.

Wu, J. (2017), ‘Introduction to convolutional neural networks’, *National Key Lab for Novel Software Technology. Nanjing University. China* **5**(23), 495.

Xie, J., Yan, W., Mu, C., Liu, T., Li, P. & Yan, S. (2016), ‘Recognizing violent activity without decoding video streams’, *Optik* **127**(2), 795–801.
URL: *https://www.sciencedirect.com/science/article/pii/S0030402615015338*

Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., Liu, Y., Zhao, Y., Pei, D., Feng, Y. et al. (2018), Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications, *in* ‘Proceedings of the 2018 world wide web conference’, pp. 187–196.

Yadav, P., Regundwar, P., Wyawahare, A., Pawar, P. & Madake, J. (2020), An intelligent system to detect violent mob activities, *in* ‘2020 IEEE 17th India Council International Conference (INDICON)’, IEEE, pp. 1–10.

Yang, W., Wang, R. & Wang, B. (2020), Detection of anomaly stock price based on time series deep learning models, *in* ‘2020 Management Science Informatization and Economic Innovation Development Conference (MSIEID)’, IEEE, pp. 110–114.