EXTREME ACTION RECOGNITION FROM REAL-TIME VIDEO USING TIME-SERIES DEEP LEARNING MODEL

GOH QING HAO

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2021

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	GOH QING HAO
ID No.	:	17UEB00099
Date	:	25th September 2021

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"EXTREME ACTION RECOGNITION FROM REAL-TIME VIDEO USING TIME-SERIES DEEP LEARNING MODEL"** was prepared by **GOH QING HAO** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Ir. Ts. Dr. Hum Yan Chai
Date	:	25th September 2021

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor for the guidance and motivation he has given me throughout this Final Year Project and UTAR for providing me with the opportunity to take part in the Final Year Project. My most profound appreciation to my parents, who have encouraged me and funded this research.

ABSTRACT

The development of an extreme action recognition model to automate police surveillance can improve police deployment speed to crime scenes such as assault, robbery, kidnapping and other offences. However, the existing solution of extreme action recognition is insufficient to be deployed with high confidence. This study proposed a time-series deep learning model to perform extreme action recognition, built with an efficient dual streams Convolutional Neural Network integrating with Convolutional Long-Short Term Memory. Notably, a novel attempt to employ background-subtracted pose keypoints as input for the recognition. Furthermore, the proposed method demonstrated improved background noise resistance when tested in the datasets of Hockey, Movies, Violent-Flow, and RWF-2000. As a result, the ablation study shows that complementing the RGB frame difference with pose keypoints will improve the framework's accuracy. The performance of the proposed framework is comparable to the existing state-of-the-arts on the RWF-2000 dataset at 87.00% accuracy, 100% accuracy on the Movie dataset, 97.00% accuracy on the Hockey dataset, and Violent-Flows dataset at 92% accuracy. The findings discovered in this study hold enormous potential to advance the current framework of extreme action recognition.

TABLE OF CONTENTS

DECLARATION **i**

APPROVAL FOR SUBMISSION **ii**

ACKNOWLEDGEMENTS **iv**

ABSTRACT **v**

TABLE OF CONTENTS **vi**

LIST OF TABLES **viii**

LIST OF FIGURES **ix**

LIST OF SYMBOLS / ABBREVIATIONS **xiii**

CHAPTER

1 INTRODUCTION **1**

1.1 Introduction to Extreme Action Recognition 1

1.2 Importance of the Study 1

1.3 Problem Statement 2

1.4 Aim and Objectives 3

1.5 Scope and Limitation of the Study 3

1.6 Contribution of the Study 4

1.7 Outline of the Report 4

2 LITERATURE REVIEW **5**

2.1 Extreme Action Dataset 5

2.1.1 Movies Fight Dataset 5

2.1.2 Hockey Fight Dataset 6

2.1.3 Violent-Flows Dataset 6

2.1.4 UCF-101 Dataset 7

2.1.5 RWF-2000 dataset 8

2.1.6 Summary of datasets 8

2.2 Artificial Neuron and Neural Network 9

2.2.1 Artificial Neuron 9

2.2.2 Feedforward Neural Network 11

2.2.3 2D Convolutional Neural Network 12

2.3 Transfer Learning 15
2.4 Time-series Network 16
2.4.1 3D Convolutional Neural Network 16
2.4.2 Long short-term memory 17
2.4.3 Convolutional LSTM 19
2.5 Extreme Action Recognition 20
2.5.1 Recognition with SVM as classifier 20
2.5.2 Recognition with 3D CNN as classifier 23
2.5.3 Recognition with ConvLSTM as classifier 28
2.5.4 Motivation 29
3 METHODOLOGY AND WORK PLAN 32
3.1 Introduction 32
3.2 Dataset to be used for framework training 33
3.3 Proposed framework 33
3.3.1 Input of the framework 33
3.3.2 Pose estimation 34
3.3.3 Background subtraction 37
3.3.4 2D CNN feature extractor 38
3.3.5 ConvLSTM spatio-temporal encoder 39
3.3.6 Classification 40
3.3.7 Overall framework 41
3.4 Experiment and ablation study 42
3.5 Development environment 42
RESULTS AND DISCUSSION 44
4.1 Quantitative Results 44
4.1.1 Accuracy 44
4.1.2 Training analysis 45
4.1.3 Training time 46
4.2 Qualitative Results 47
4.3 Summary of critical findings 53
5.1 Conclusions 54
5.2 Recommended Solutions 54
REFERENCES 56

LIST OF TABLES

Table 2.1: Table of datasets available that is dedicated to development of extreme action time-series model. 9

Table 2.2: Comparison of Accuracy for Reviewed Methodologies. 30

Table 3.1: Table of available options for pose estimation model tested on the COCO test-dev dataset. (Güler, Neverova and Kokkinos, 2018; Cao *et al.*, 2021; Zhang *et al.*, 2021) 35

Table 3.2: Tables of object detection framework showing available options for open-source object detection frameworks (Bochkovskiy, Wang and Liao, 2020) 36

Table 3.3: Table of open-source options of 2D CNN that can be used for feature extraction. (PyTorch, no date; Howard *et al.*, 2019; Véstias, 2019) 38

Table 3.4: Architecture of Classification layers for the proposed framework. 40

Table 3.5: Specification of hardware for deep learning. 43

Table 3.6: Two popular options of cloud computing solutions showing the estimated monthly cost, the GPU availability, and the runtime limit. (Google Colab, no date) 43

Table 4.1: Accuracy obtained with proposed framework and ablation study compared to major benchmarks from previous studies. 44

Table 4.2: Result obtained from ablation study. 45

LIST OF FIGURES

Figure 1.1: Crime index ratio of each states in Malaysia, calculated per 100,000 population ratios. (Mahidin, 2020) 2

Figure 2.1: Sample images of fighting scenes extracted from movies dataset. (Gracia et al., 2015) 5

Figure 2.2: Samples images of fighting scene from the hockey fight dataset. (Gracia et al., 2015) 6

Figure 2.3: Sample of images from violent flows dataset depicting scenes of crowd activities and crowd violence. (Hassner, Itcher and Kliper-Gross, 2012) 7

Figure 2.4: Sample images of punching and rafting labels from UCF-101 dataset. (Soomro, Zamir and Shah, 2012) 7

Figure 2.5: Sample of images from RWF-2000 dataset depicting scenes of retrieved from CCTV footages. (Cheng, Cai and Li, 2019) 8

Figure 2.6: Graphical representation of human neuron anatomy. (Freudenrich and Boyd, 2021) 10

Figure 2.7: Schematics of a basic artificial neuron. (Rosebrock, 2019) 11

Figure 2.8: Example of feedforward neural network. (Rosebrock, 2019) 11

Figure 2.9: Schematics of feedforward neural network for MNIST numeric digit prediction. (Github, 2021) 12

Figure 2.10: Basic structure of CNN for multi-class classification with feature learning layers and classification layers. (Matlab, 2021) 13

Figure 2.11: Graphical depiction of convolution function. (IBM Cloud Education, 2020) 13

Figure 2.12: Graphic depiction of max pooling operation with 2x2 kernel and stride of two (Rosebrock, 2019) 14

Figure 2.13: Schematics of AlexNet CNN architecture. (Wayne, 2020) 15

Figure 2.14: Transfer Learning, Pre-trained Model Utilised as Fixed Feature Extractor. (Oquab et al., 2014) 16

Figure 2.15: Graphical depiction of 3D CNN convolution repeated across time. (Ji et al., 2013) 17

Figure 2.16: Schematics of a single LSTM memory cell. (Yu et al., 2019) 19

Figure 2.17: The convolution of input across sequence of time-steps occurring at the input of ConvLSTM. (Shi et al., 2015) 20

Figure 2.18: Model Architecture of Multi-Stream Learning. (Carneiro et al., 2019) 22

Figure 2.19: Example of human pose information extracted in the study with OpenPose. (Nova, Ferreira and Cortez, 2019) 23

Figure 2.20: Model Architecture of method SSD + 3D CNN. (Ullah et al., 2019) 24

Figure 2.21: Graph of epochs plotted against loss, showing long period of no improvements after 1000 epochs. (Ullah et al., 2019) 25

Figure 2.22: Model Architecture of method SSD combined with Optical Flow and 3D CNN, the recognition framework is separated into localisation branch and recognition branch. (Xu, See and Lin, 2019) 26

Figure 2.23: The model architecture of end-to-end 3D CNN capable of recognition without pre-processing steps. (Li et al., 2019) 26

Figure 2.24: Model architecture of densePose RCNN combined with 3D CNN. (Calzavara, 2020) 27

Figure 2.25: Model architecture of 2D CNN combined with ConvLSTM. (Sudhakaran and Lanz, 2017) 28

Figure 2.26: Model architecture of dual-stream 2D CNN + Separable ConvLSTM. (Islam et al., 2021) 29

Figure 3.1:	Initial plan to develop the extreme action re framework, each box illustrates the process involved.	32
Figure 3.2:	The image shown on the left is the original video frame before pose estimation; the image on the right shows pose keypoints of the pedestrians drawn on the black canvas after pose estimation.	37
Figure 3.3:	Graphical model of convolutional long-short term memory cell.	40
Figure 3.4:	The proposed framework of this project showing all significant components.	41
Figure 4.1:	Graph of loss and accuracy for method of RGB + EfficientPose C on all datasets during training, plotted against epoch.	46
Figure 4.2:	Training time taken per epoch for different method training on RWF-2000 dataset.	47
Figure 4.3:	False positive case with method RGB+EfficientPose C with sample from hockey dataset showing incorrect pose estimation	48
Figure 4.4:	False positive case with method RGB+EfficientPose C with sample from RWF-2000. The pose estimation correctly predicted the pose keypoints in most cases.	49
Figure 4.5:	Dispersed red colour static noises observed from the RGB frame difference in the false positive case with method RGB+EfficientPose C with sample from RWF-2000.	49
Figure 4.6:	False positive case with method RGB+EfficientPose C with test sample from RWF-2000, drastic scene change observed at the final frame.	50
Figure 4.7:	False negative case with method RGB+EfficientPose C with sample from RWF-2000 dataset depicting correct pose estimation, but subjects of interest were too small.	51

Figure 4.8: False negative case with method RGB+EfficientPose C with sample from Violent-Flows crowd dataset depicting incorrect pose estimation and excessive noise in RGB frame difference. 52

LIST OF SYMBOLS / ABBREVIATIONS

f	forget gate
i	gate component
$\tilde{c}$	memory cell component
o	output gate
c	cell state
h	hidden state
t	time steps
x	input vector
I	image input vector
w	weight vector
b	bias vector
σ	sigmoid function
$tanh$	hyperbolic tangent function
$*$	convolution function
n	sequence of {1,…,N}
L	total number of frames in video
N	sequence length
i	index of frame to be sampled
1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
AI	artificial intelligence
CCTV	closed-circuit Television
CNN	convolutional neural network
CONV	convolutional
ConvLSTM	convolutional long short-term memory
CPU	central processing unit
FC	fully connected
FPS	frames per second

GPU	graphics processing unit
MMACC	million multiply-accumulation
mAP	mean average precision
mAR	mean average recall
OS	operating system
PL	pooling
RAM	random access memory
RGB	red, green, blue
RNN	recurrent neural network
ROI	region of interest
SVM	support vector machine
SOTA	state of the art
SSD	single shot setector
VRAM	video random-access memory

CHAPTER 1

INTRODUCTION

1.1 Introduction to Extreme Action Recognition

Extreme action recognition is an artificial intelligence (AI) framework to monitor incidents of activities such as fighting through closed-circuit television (CCTV) surveillance footage. When an extreme action is detected in a particular place, police and ambulance can be notified to intervene and help to prevent injuries or death.

Most extreme action recognition is still performed manually, with security guards monitoring an array of CCTV surveillance footage. However, manual vision monitoring can be unreliable because it is dull labour and requires a high level of focus and attention, prone to failure when the person monitoring the CCTV surveillance footage becomes distracted or drowsy. Hence, developing a powerful action recognition solution assisted or fully automated with AI would significantly improve surveillance efficiency.

Extreme action recognition from real-time video footage using AI is still an emerging technology; thus, no reliable solution exists. Computer science researchers have been developing an AI model architecture dedicated to performing reliable extreme action recognition; nevertheless, the existing solution of extreme action recognition is insufficient to be deployed with high confidence.

1.2 Importance of the Study

The findings of this study will ultimately improve public safety through contribution to the development of an extreme action recognition framework. For instance, public safety can be improved by equipping police authorities with AI-assisted extreme action recognition; the recognition framework deployed at crime hotspots can automatically alert the police force of incidences of robbery and assault. Early detection of crime leads to rapid deployment of the police force, increasing effectiveness in preventing bodily injuries and death.

1.3 Problem Statement

The rapid urbanisation of Malaysia has led to an increased frequency of street crimes; the most urbanised states of Kuala Lumpur, Johor, Selangor, and Pinang together contributed 70% of all street crimes occurring in Malaysia. In 2010, the government of Malaysia proposed the Government Transformation Programme roadmap targeted to reduce the crime rate in the urban area through the installation of surveillance cameras monitored by the Royal Malaysia Police. (Soh, 2012)

However, by 2019 the urbanised states of Kuala Lumpur, Selangor, Sembilan, Pinang, Melaka, and Johor continue to display a high crime index ratio by ranking higher than the remaining less urbanised states, as shown in Figure 1.1. Notably, the crime index in the capital region of Kuala Lumpur leads the national average by 130% despite the installation of 40,000 surveillance cameras in the highly urbanised Kuala Lumpur. (Adilah, 2017)

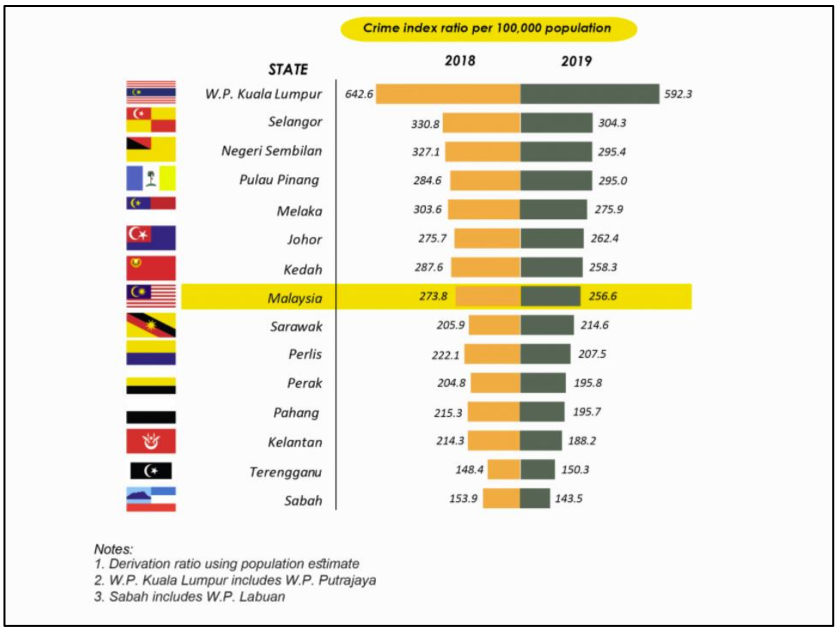

Figure 1.1: Crime index ratio of each states in Malaysia, calculated per 100,000 population ratios. (Mahidin, 2020)

The increased number of surveillance cameras does not guarantee crime reduction; an effective surveillance system requires constant monitoring. The installed cameras are mostly hidden and monitored only upon the report of a crime. As a result, the deterrence capability of surveillance cameras is minimised. The adaptation of AI-assisted extreme action recognition that actively monitors the camera and alerts the police in case of anomaly may significantly improve the effectiveness of surveillance in crime deterrence and criminal apprehension.

1.4 Aim and Objectives

This project will aim to develop an extreme action recognition based on deep learning time-series model that can recognise extreme action from video clips. The objectives of this project to accomplish the aim are:

(i) Conduct literature reviews of extreme action recognition.

(ii) Identify suitable datasets for training and performance benchmarks.

(iii) Develop the extreme action recognition framework with deep learning.

(iv) Train and test the developed model.

(v) Evaluate the test results and identify the root cause of failure for the developed model.

(vi) Proposed suggestions for improvement.

1.5 Scope and Limitation of the Study

This project's scope is to develop a method to perform extreme action recognition trained with some popular benchmarks. The use of popular benchmarks allows the developed method to be compared to other related works.

The reliability of the extreme action recognition AI model depends on the quality, diversity, and volume of the dataset. Unlike the ImageNet dataset dedicated to training object recognition, which contains over 1.2 million static labelled images, the labelled large-scale dataset for extreme action is limited. Hence, one of the limitations of this study is the availability of an extreme action dataset.

Project time constraint is also a limitation. AI deep learning computation requires hours for a single epoch. Typically, thousands of epochs are needed to train a model at the optimal learning rate fully. Thus, a suitable number of epochs and learning rate was chosen to reduce training time to ensure completion of the project within the available project time.

1.6 Contribution of the Study

The existing extreme action recognition frameworks state of the art (SOTA) exaggerate background noises when background subtraction is computed on unprocessed input. This study proposed and evaluated a novel method to improve noise resistance by complementing unprocessed input with pose keypoints using dual streams convolutional neural network (CNN) integrated with convolutional long-short term memory (ConvLSTM).

In addition, this work implemented EfficientPose to give a pose keypoints with improved mean average precision and recall compared to DensePose and OpenPose used by previous works. Three experiments were conducted with different versions of EfficientPose to evaluate the correlation between pose estimation accuracy and recognition accuracy. Lastly, an ablation study was performed to determine the effectiveness of combining pose keypoints with ConvLSTM. The combination of pose keypoints with ConvLSTM is the first attempt in literature at the time of writing.

1.7 Outline of the Report

This report presented a critical review of existing extreme action recognition solutions, methodology to implement the proposed framework, and analysis of results obtained with the proposed framework. Chapter 1 introduced the potential of an automated police surveillance system with extreme action recognition and the problem statement on the lack of an existing solution. Finally, the aims and objectives to develop a solution were written, along with the scientific community's scope, limitation, and contribution.

Chapter 2 presented the study's motivations based on the limitations found in the literature reviews on existing extreme action recognition datasets and solutions. The existing solutions reviewed were categorised into three groups based on different classifiers: Support Vector Machine (SVM), Three-Dimensional (3D) CNN, and ConvLSTM.

The methodology to develop the solution to achieve the aims and objectives of this study was described in chapter 3; the results obtained with the proposed method was analysed in chapter 4. Lastly, chapter 5 concludes the critical findings observed in this project and recommends potential improvements in future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Extreme Action Dataset

In training an AI model for extreme action recognition, datasets consisting of video clips of extreme action is essential. This sub-section reviews the open-access extreme action dataset available on the internet.

2.1.1 Movies Fight Dataset

Gracia et al. (2015) introduced the Movies dataset consisting of video clips extracted from action movies and 200 video clips; each of the 200 videos is labelled either as “Fight” or “Non-fight” and the dataset is evenly split between the two labels. Each video clip consists of 50 video frames, resized to a resolution of 720x576 and played at 25 frames per second (FPS). The videos in this dataset have distinctly different scales, backgrounds, and content. Figure 2.1 shows four sample images of this dataset.

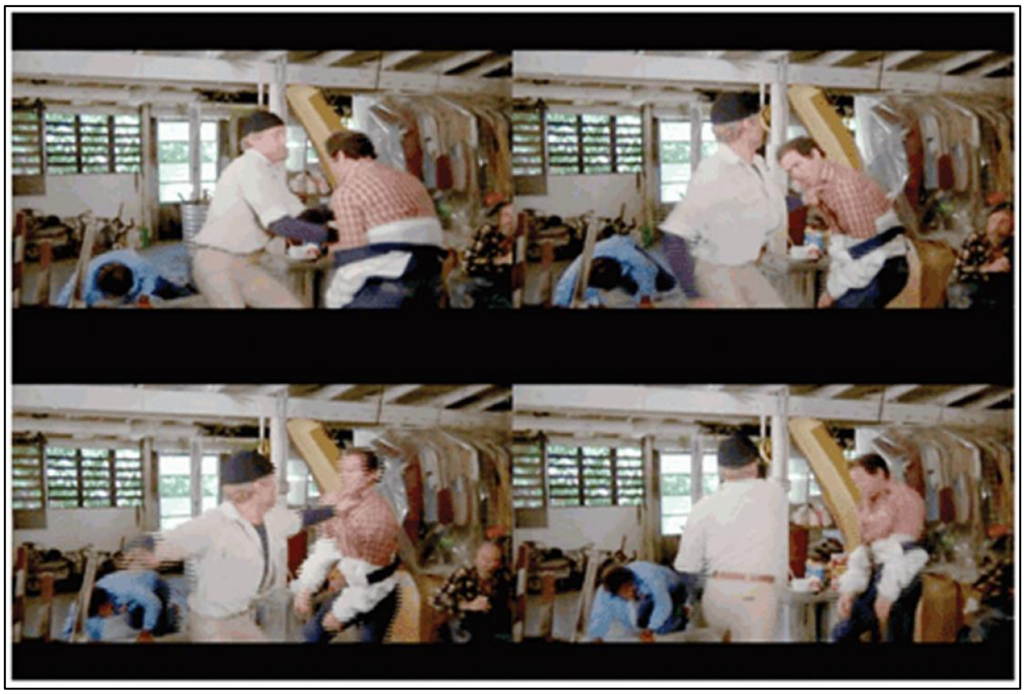

Figure 2.1: Sample images of fighting scenes extracted from movies dataset. (Gracia et al., 2015)

2.1.2 Hockey Fight Dataset

The Hockey dataset was published along with the Movies dataset, consisting of 1000 video segments clipped from the broadcast of hockey games from the National Hockey League; each of the 1000 videos is labelled either as “Fight” or “Non-fight”, and the dataset is evenly split between the two labels. Each video clip consists of 50 video frames, resized to a resolution of 320x240 and played at 25 FPS. The videos in this dataset have identical scales, backgrounds, and content. Figure 2.2 shows four sample images of this dataset. (Gracia *et al.*, 2015)

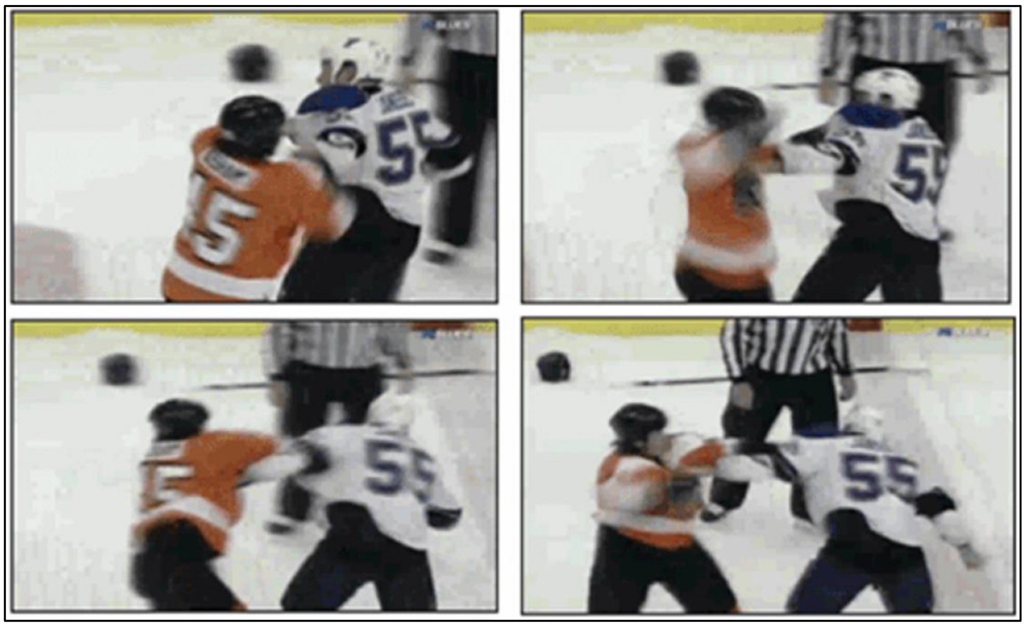

Figure 2.2: Samples images of a fighting scene from the hockey fight dataset. (Gracia et al., 2015)

2.1.3 Violent-Flows Dataset

Published by Hassner, Itcher and Kliper-Gross (2012), the Violent-Flows dataset is dedicated to benchmarking extreme action recognition among a crowd of people. The dataset consists of 246 video clips evenly split between two labels of “Violence” and “Non-Violence”. Each video clip has a varying length of video frames, a fixed resolution of 320x240 and played at either 25 FPS or 30 FPS. Figure 2.3 shows four sample images of this dataset.

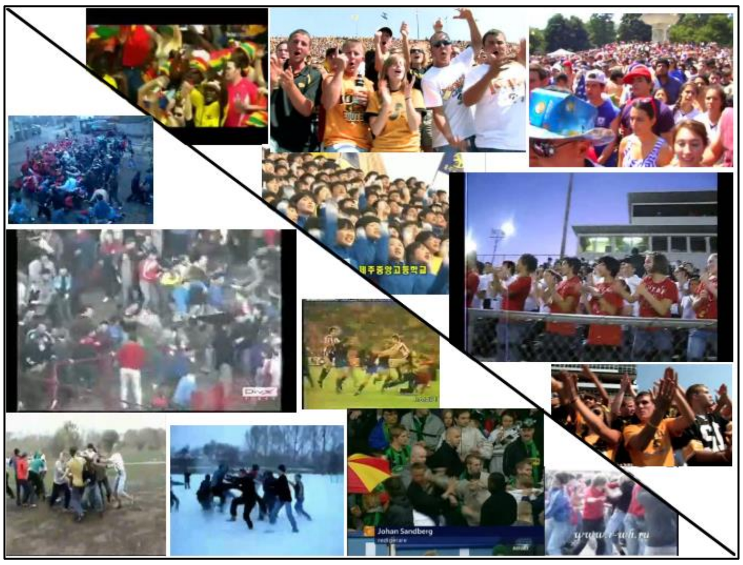

Figure 2.3: Sample of images from violent flows dataset depicting scenes of crowd activities and crowd violence. (Hassner, Itcher and Kliper-Gross, 2012)

2.1.4 UCF-101 Dataset

The UCF-101 dataset is a large-scale human action class dataset consisting of video clips extracted from YouTube and labelled at the video level, and it consists of 13,320 video clips split across 101 different class labels. Each video clip has a varying length of video frames, a fixed resolution of 320x240 pixels, and a fixed 25 FPS. This dataset has a broad spectrum of labels, including actions such as walking, running, biking, boxing, and so forth. Most labels are related to sports and regular human activity; however, a few labels such as punching, sumo wrestling, and fencing may be considered a form of extreme action. Figure 2.4 shows two sample images of this dataset. (Soomro, Zamir and Shah, 2012)

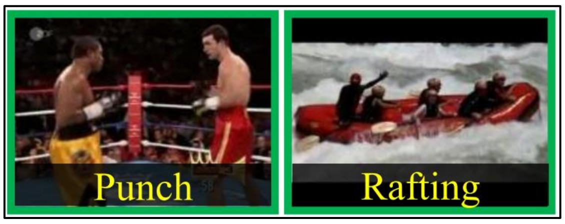

Figure 2.4: Sample images of punching and rafting labels from the UCF-101 dataset. (Soomro, Zamir and Shah, 2012)

2.1.5 RWF-2000 dataset

Cheng, Cai and Li (2019) published the RWF-200 dataset, a large-scale extreme action dataset containing 2000 video clips extracted from YouTube. The 2,000 video clips are evenly split between two labels of “Fight” and “Non-Fight” with a pre-defined train-test split ratio of 80:20. Each video clip has a fixed time length of 5 seconds with varying resolutions. The videos in this dataset are all surveillance-based footage with slightly elevated viewpoints, and each clip has a distinctly different scale, FPS, background, and content. Figure 2.5 shows several sample images from the RWF-2000 dataset.

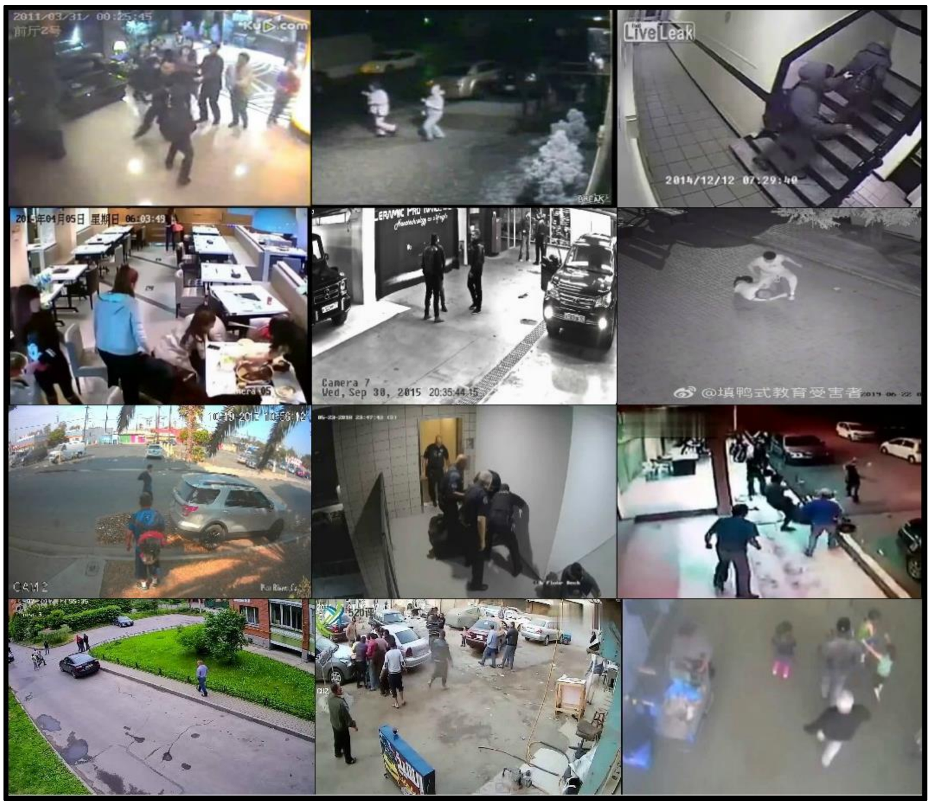

Figure 2.5: Sample of images from RWF-2000 dataset depicting scenes retrieved from CCTV footages. (Cheng, Cai and Li, 2019)

2.1.6 Summary of datasets

Table 2.1 summarises the datasets reviewed in sub-chapter 2.1. Among the five datasets, UCF-101 has the greatest number of videos. However, among the 101 different labels, there were no consensus or guidelines of what should be considered extreme action. Therefore, different existing research that evaluated their framework on UCF-101 cannot be compared.

Table 2.1: Table of datasets available that is dedicated to the development of extreme action time-series model.

Datasets	**Description**	**Number of Videos**	**Number of labels**
UCF-101	Broad spectrum of human activity	13,320	101
Violent-Flows	Violent and non-violent activity in crowd	246	2
Hockey	Violent and non-violent activity in National Hockey League broadcast	1000	2
Movies	Violent and non-violent activity in Hollywood movies	200	2
RWF-2000	Violent and non-violent activity recorded in surveillance footages	2000	2

This study has found that the Movies, Hockey, Violent-Flows, RWF-2000 datasets have been used in existing literature as the benchmarks of recognition performance. These four datasets were chosen as they have simple true/false label to differentiate whether the action is extreme.

2.2 Artificial Neuron and Neural Network

This section of the literature review briefly reviews the basic concepts of artificial intelligence utilised in this project; in subsections 2.2.1 and 2.2.2, artificial neurons and feedforward neural networks are discussed. In 2.2.3, the various building blocks of CNN and the AlexNet variant of CNN were reviewed.

2.2.1 Artificial Neuron

As shown in Figure 2.6, brain neuron cell consists of a cell body, dendrites, axons, and synapses. The human brain consists of many neurons interconnected to each other, wherein synapses of a neuron are connected to the dendrites of another neuron. Hence, a complex neural pathway is created. The dendrites can be thought of as the input, synapses the output, and cell body the decision-maker. Each neuron receives electrochemical signals from other neurons through its dendrites, and if the aggregation of the electric potential of these signals is strong enough, the cell body becomes activated and will transmit signals to other neurons. (Rosebrock, 2019; Freudenrich and Boyd, 2021)

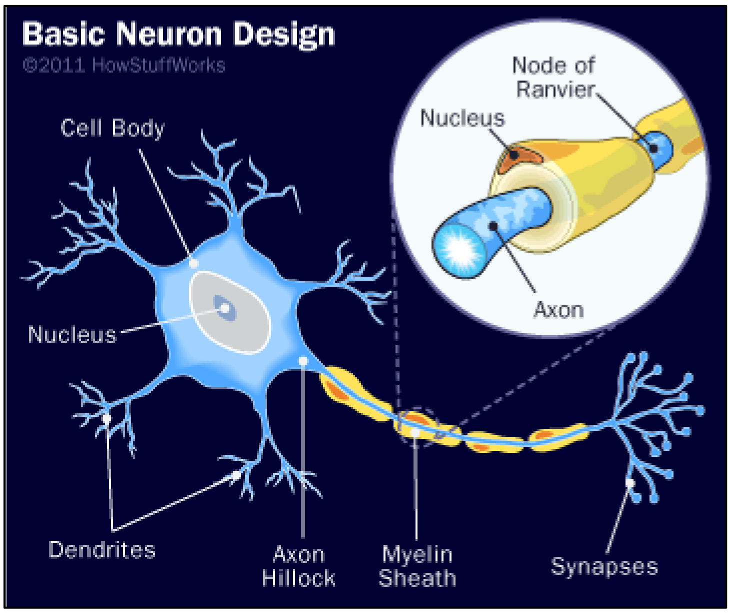

Figure 2.6: Graphical representation of human neuron anatomy. (Freudenrich and Boyd, 2021)

An artificial neuron is an attempt to artificially create intelligence using a modern computing system inspired by a neuron's working principles in a brain. A schematic of the artificial neuron, as illustrated in Figure 2.7, consists of inputs, weights, weighted sum, transfer function, and output. The input and output are analogous to the dendrite and synapses, respectively, whereas the weights, weighted sum, and transfer function form the decision-maker, analogous to the cell body. When an artificial neuron receives inputs, the value of each input is multiplied with the respective weights and summed together. The transfer function, in this case, a step function, will become activated and sends a signal to the output if the weighted sum exceeds a certain threshold; otherwise, the step function remains inactivated, and no signal will be sent. (Suzuki, 2011)

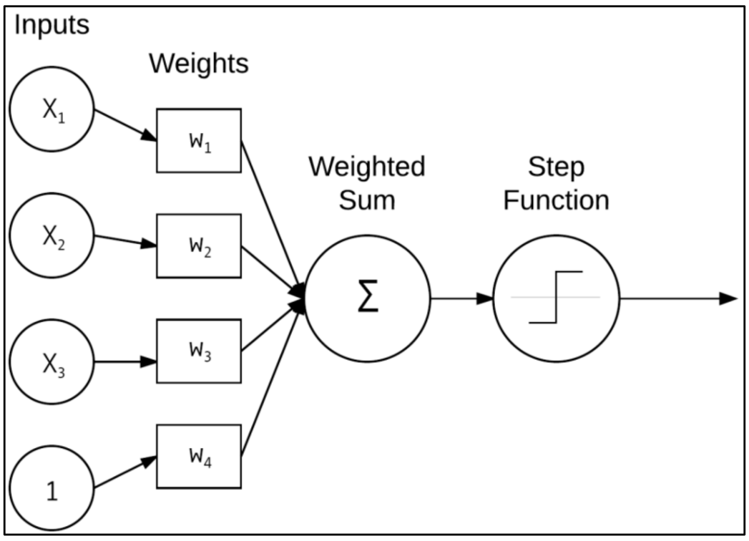

Figure 2.7: Schematics of a basic artificial neuron. (Rosebrock, 2019)

2.2.2 Feedforward Neural Network

A feedforward neural network can be created by interconnecting artificial neurons with each other in a layer as shown in Figure 2.8, whereby the first layer of neurons is connected to the input, and they can be activated depending on the input data, and in turn, output their signals to next layer of neurons. The cycle repeats until the signal is propagated to the output layer, which gives the prediction. (Rosebrock, 2019)

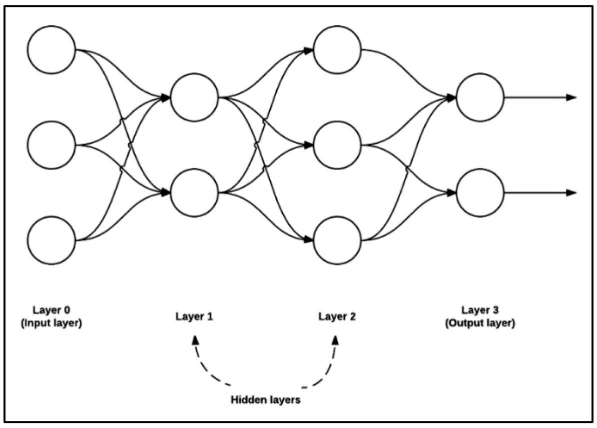

Figure 2.8: Example of a feedforward neural network. (Rosebrock, 2019)

An example of a feedforward neural network in image-based prediction is the prediction of a numeric digit based on images from the MNIST database. As shown in Figure 2.9, the feedforward neural network input layer consists of 784 inputs, each corresponding to a pixel on the input image. The first layer in the network consists of 10 artificial neurons, in which each artificial neuron can become activated depending on the type of input. The output layer also consists of 10 artificial neurons that correspond to 10 digits from 0 to 9. Prediction occurs at the output layer, whereby only one of the output neurons will be activated. For example, in Figure 2.9, when an image of 8 was put into the network, the expected result is that only the output neuron representing eight will become activated. (Github, 2021)

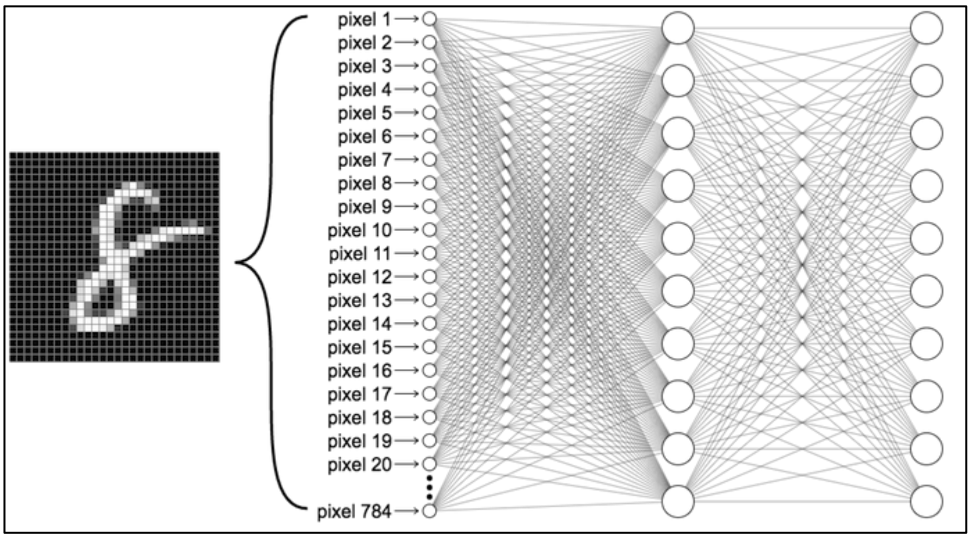

Figure 2.9: Schematics of feedforward neural network for MNIST numeric digit prediction. (Github, 2021)

2.2.3 2D Convolutional Neural Network

Two-dimensional (2D) CNN is a class of neural networks that is consisted of more components compared to the feedforward neural network introduced in sub-chapter 2.2.2. The components of a typical 2D CNN shown in Figure 2.10 are separated into two sections: the Feature Learning section with alternating convolutional (CONV) and pooling (PL) layers and the Classification Section with multiple fully-connected (FC) layers. (Matlab, 2021)

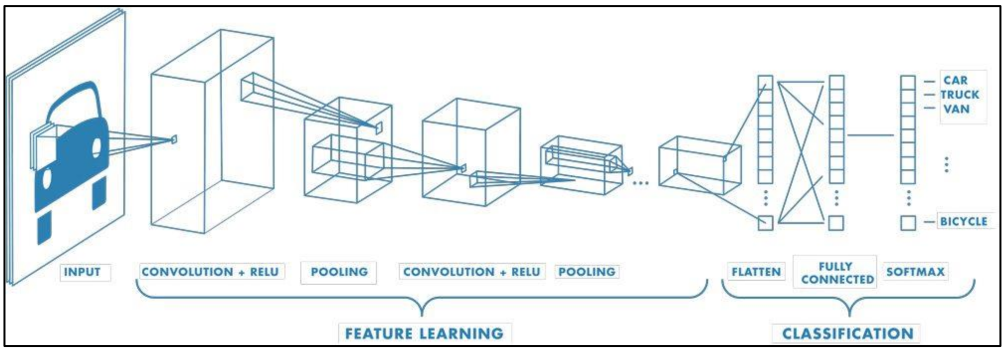

Figure 2.10:Basic structure of CNN for multi-class classification with feature learning layers and classification layers. (Matlab, 2021)

The CONV layer is a set of filters that convolve every region of the image. Each filter is connected to a small local region of the input image, and multiple filters are used to ensure every region of the input image is connected to a filter. If the image has a depth of three, then three different filters will be connected to the same region. The output of each filter in the CONV layer is individually connected to an artificial neuron, which is analogous to other kernel filters such as the average filter. The weights of neurons input can be changed based on the deep learning process, whereas the weights of the kernel filter are fixed. Figure 2.11 depicts the operation of the CONV layer with regards to an input image, wherein the values inside the filter are equivalent to the weights of the artificial neuron. (Wood, 2021; IBM Cloud Education, 2020)

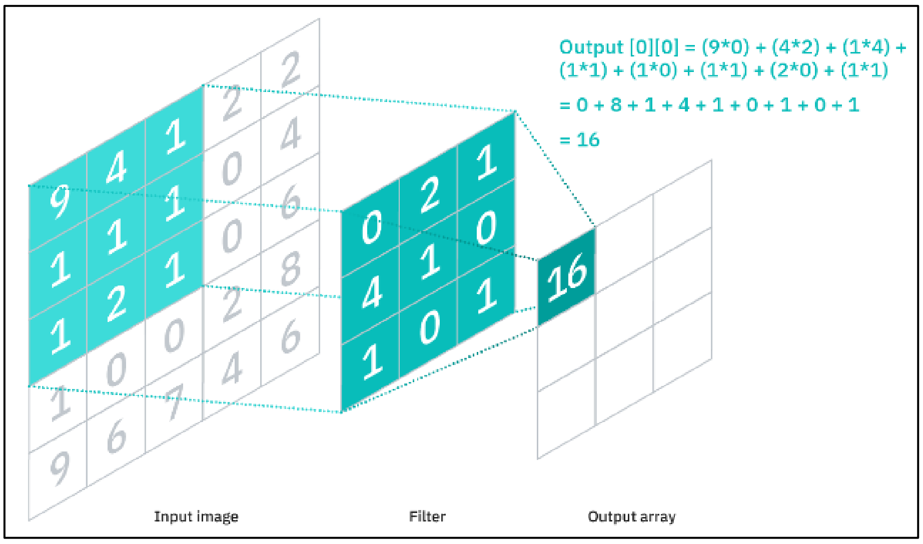
Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1*0) + (1*1) + (2*0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16
Input image
Filter
Output array

Figure 2.11:Graphical depiction of convolution function. (IBM Cloud Education, 2020)

The PL layer is applied to reduce the spatial size of the input without losing information or context from the input. For example, the PL layer in Figure 2.12 sub-divides the input into the smaller region and retains only the maximum value within the sub-region. Implementing the Pooling layer is essential to reduce the number of inputs, which, in turn, significantly reduces computer resource requirements. (Rosebrock, 2019)

Finally, the FC layer is used as the prediction and output layer. The principle of the FC layer works the same as the feedforward neural network introduced in sub-chapter 2.2.2, whereby the only difference is that the input layer is connected to the output of the pooling layer instead of pixels on the image. (Rosebrock, 2019)

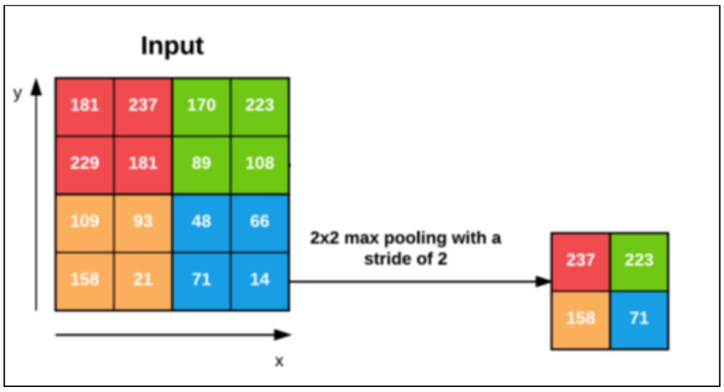

Figure 2.12: Graphic depiction of max pooling operation with 2x2 kernel and stride of two (Rosebrock, 2019)

One famous example is the AlexNet CNN Architecture designed and initially proposed by Krizhevsky, Sutskever and Hinton (2012) that won the 2012 ImageNet competition. As shown in Figure 2.13, the model architecture is separated into two sections: the ConvNet Feature Extractor and the Classification Head. The ConvNet Feature Extractor consists of alternating CONV and PL layers that extract local features from the input image. The Classification Head consists of only FC layers to perform the prediction and output. As a result, the AlexNet CNN can predict 1000 different classes of images ranging from different types of animals, fruits, cars, and ships with an error rate of only 15.3%.

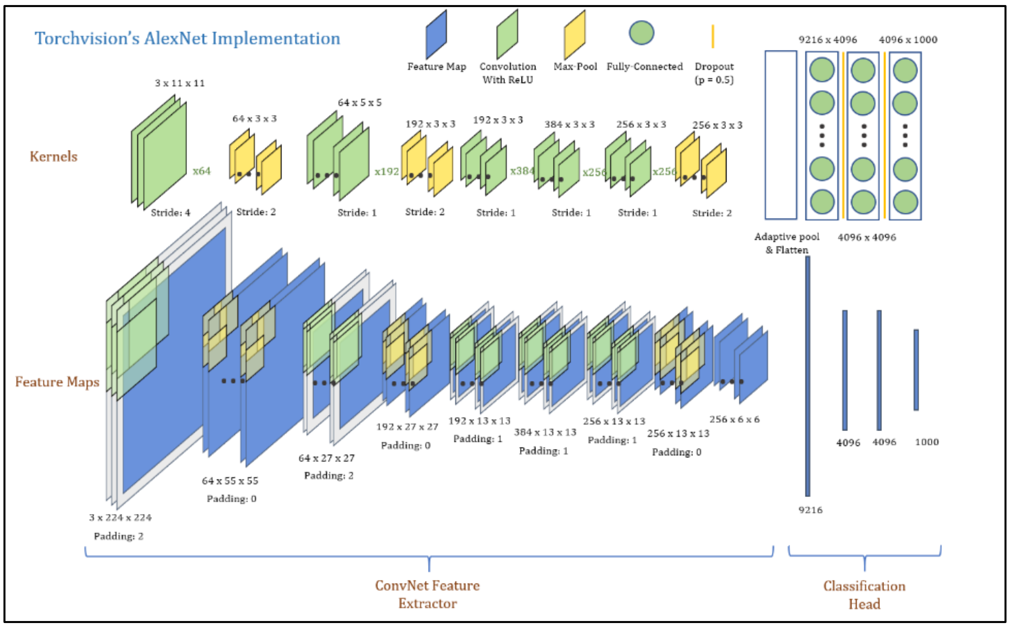

Figure 2.13: Schematics of the AlexNet CNN architecture. (Wayne, 2020)

2.3 Transfer Learning

Transfer learning is a deep learning method of transferring the model that had learnt to solve a certain problem, to become the starting point for a new deep learning process to solve a new and different problem. For example, the features of a model that had learnt to perform face detection can be used to create a new model in performing facial recognition. In addition, transfer learning is used to train models in insufficient data available or limited computational resources. (Chollet, 2020; Chilamkurthy, 2017)

There are two common approaches to perform transfer learning, the first approach is to fine-tune the pre-trained model to adapt to solve a new problem in the same target environment. The pre-trained model architecture remains unchanged, and the model is fine-tuned with additional data, such that the model becomes updated.

In the second approach, a part of the pre-trained model as a feature extractor to solve a problem in a different environment. As shown in Figure 2.14, the pre-trained model architecture is altered whereby the last few FC layers of the network are removed, and new adaptation FC layers of a different design with randomly initialised weights are then attached, and the modified architecture is trained on the new target tasks. (Oquab et al., 2014; Chilamkurthy, 2017)

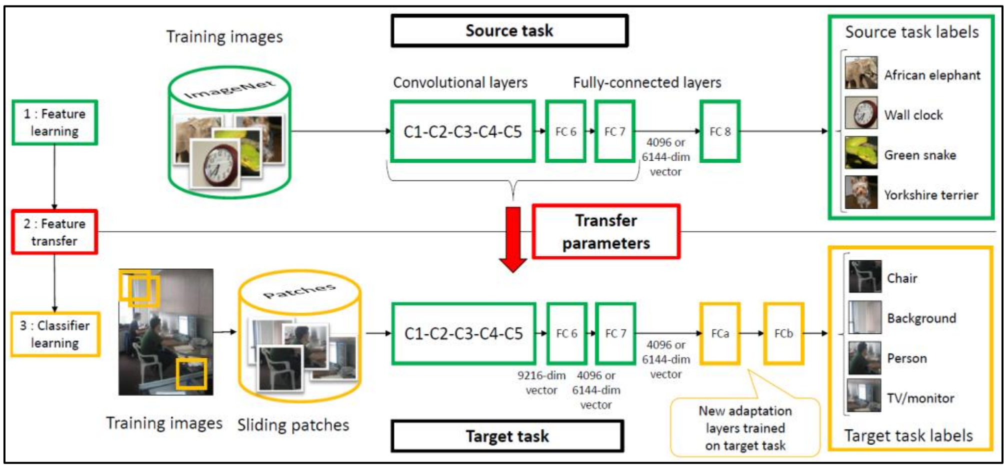

Figure 2.14: Transfer learning of a pre-trained model utilised as a fixed feature extractor. (Oquab et al., 2014)

2.4 Time-series Network

Time-series problems such as weather forecasting, action recognition, stock market prediction are common but are highly complex and challenging. The AIs are expected to make predictions based on aggregated information extracted from every given time frame.

The current AI architectures, such as the AlexNet 2D CNN introduced in sub-chapter 2.2.3, are limited to making predictions on a temporal-independent image within the scope of computer vision. For example, the AI can predict what is in the image, the position of the person relative to the image and draw segmentation masks across different objects within the image. However, it was difficult for AI to draw inferences on the patterns in a sequence of images. In recent years, researchers have proposed several novel neural network architectures that can encode temporal features to solve time-series problems, and this chapter will briefly introduce some of these neural networks and their state of development.

2.4.1 3D Convolutional Neural Network

To date, there are numerous variants of 3D CNN implementations, one of the variants for human action recognition was proposed by Ji, Xu, Yang, and Yu (2013). The variant can encode temporal features alongside spatial features.

This work extended the 2D CNN by one additional dimension to accommodate the temporal features.

The image processing kernels of the convolution are used to extract separate feature maps in 5 channels from each frame containing gradient in the X-Y direction, optical flow in the X-Y direction, and grayscale; subsequently, as shown in Figure 2.15, the extracted feature maps are repeated across time to become a sequence which represents time-series data. The convolution is continued deeper into the network until a linear feature vector is produced, which is connected to the FC layers to make predictions. (Ji et al., 2013)

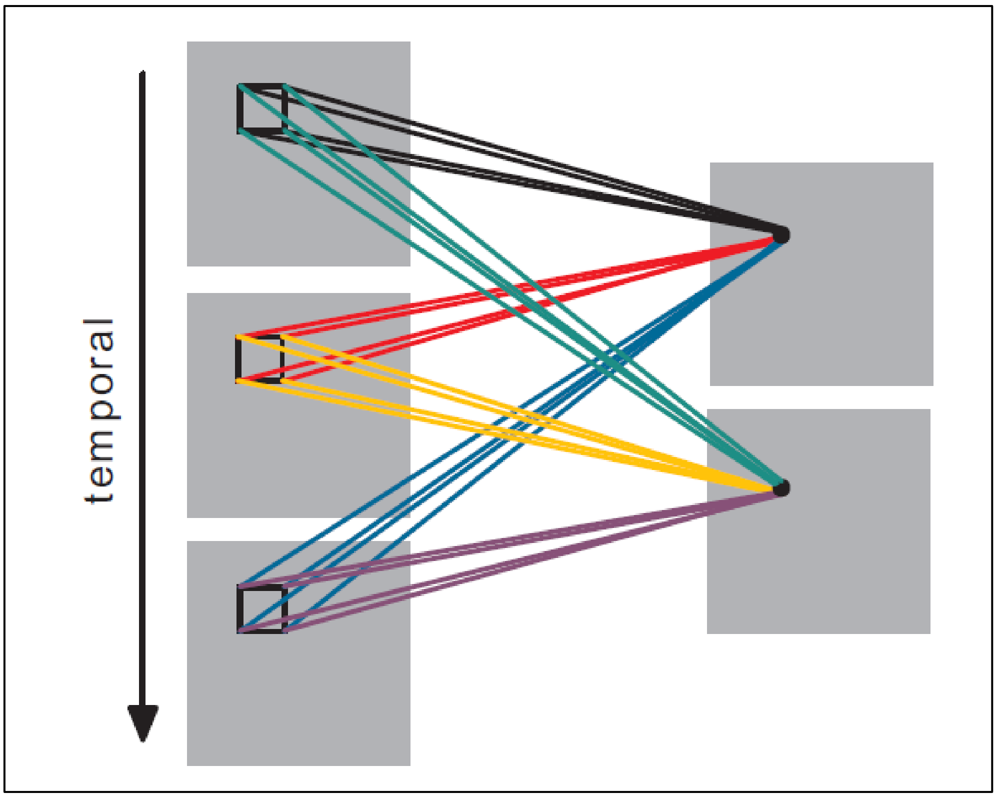

Figure 2.15: Graphical depiction of 3D CNN convolution repeated across time. (Ji et al., 2013)

2.4.2 Long short-term memory

The Long Short-Term Memory (LSTM), a variant of Recurrent Neural Network (RNN), has been used extensively to solve time-series problems such as speech recognition and translation. Proposed by Hochreiter and Schmidhuber (1997), a memory cell contains forget gate, update gate, and output gate as the three major components. The memory cell is trainable using feedforward and backpropagation like a CNN.

Each gate is represented by mathematical equations, whereby the forget gate is represented by Equation 2.1. The forget gate determines what information stored inside the cell state is no longer relevant and should be forgotten. Equation 2.2, Equation 2.3, and Equation 2.4 collectively form the input gate. The gate component and memory cell component at the input gate determine what information should be stored in the cell state. The output gate represented in Equation 2.5 and Equation 2.6 determines the output and hidden state by inferring the input and cell state. In Figure 2.16, the connections between the gates and their respective equations are illustrated. (Hochreiter and Schmidhuber, 1997; Yu et al., 2019)

$$f_t = \sigma\left(w_x^f x_t + w_h^f h_{t-1} + b^f\right) \tag{2.1}$$

$$i_t = \sigma\left(w_x^i x_t + w_h^i h_{t-1} + b^i\right) \tag{2.2}$$

$$\tilde{c}_t = tanh\left(w_x^{\tilde{c}} x_t + w_h^{\tilde{c}} h_{t-1} + b^{\tilde{c}}\right) \tag{2.3}$$

$$c_t = \tilde{c}_t i_t + c_{t-1} f_t \tag{2.4}$$

$$o_t = \sigma(w_x^o x_t + w_h^o h_{t-1} + b^o) \tag{2.5}$$

$$h_t = o_t \tanh\,(c_t) \tag{2.6}$$

where

f = forget gate

i = gate component

$\tilde{c}$ = memory cell component

o = output gate

c = cell state

h = hidden state

t = time steps

x = input vector

w = weight vector

b = bias vector

σ = sigmoid function

$tanh$ = hyperbolic tangent function

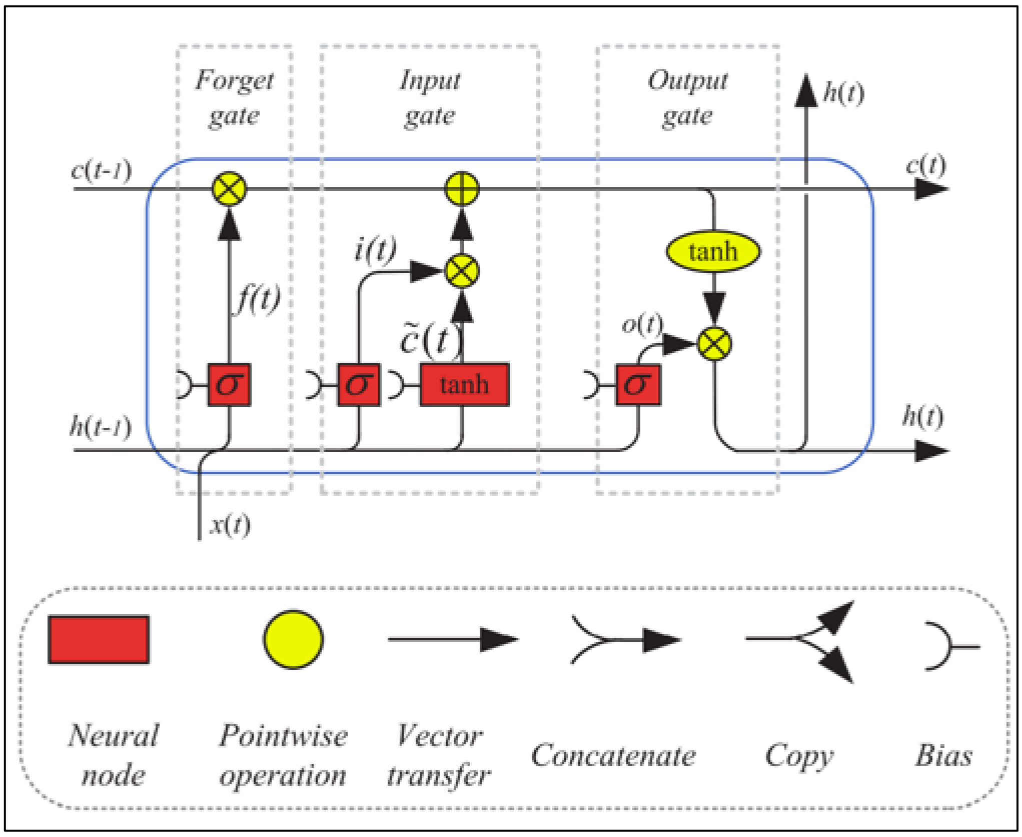

Figure 2.16: Schematics of a single LSTM memory cell. (Yu et al., 2019)

Furthermore, multiple LSTM memory cells are commonly stacked together across a sequence of input. For example, if the input window is 20-time steps, 20 LSTM memory cells would be connected. The output of one memory cell will propagate to the input of the next cell; the propagation of cell states from one cell to another act as a memory to preserve important information that is critical to perform prediction on the time-series problem. During training, the connection between LSTM cells at cell state and hidden state allows backpropagation through time, training the entire sequence together.

2.4.3 Convolutional LSTM

LSTM is limited to using a one-dimensional (1D) input vector, for example, predicting sequence from an input of text, voice segment and numbers. However, images are 2D, and LSTM cannot be used to make a prediction from a sequence of images. In order to overcome this limitation, an article by Shi, Chen, Wang, and Yeung (2015) proposed the ConvLSTM, whereby it convolves its input, x(t), output, h(t), and cell state, c(t), from 2D vector into 1D vector. The memory cell structure of ConvLSTM remains identical to LSTM, as shown in Figure 2.17. (Shi et al., 2015)

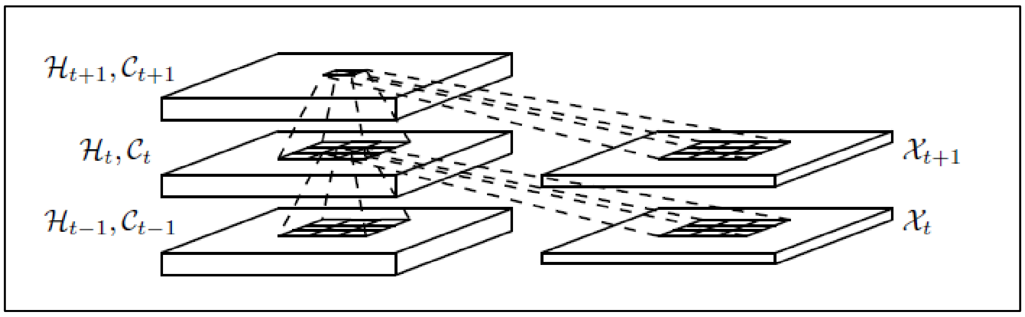
$\mathcal{H}_{t+1}, \mathcal{C}_{t+1}$
$\mathcal{H}_{t}, \mathcal{C}_{t}$
$\mathcal{H}_{t-1}, \mathcal{C}_{t-1}$
$\mathcal{X}_{t+1}$
$\mathcal{X}_{t}$

Figure 2.17: The convolution of input across sequence of time-steps occurring at the input of ConvLSTM. (Shi et al., 2015)

2.5 Extreme Action Recognition

Extreme action recognition is an AI framework that aims to recognise actions of violent events, including actions such as punching, wrestling and crowd violence. Extreme action recognition can be considered a niche sub-field of action recognition, where the framework is dedicated specifically to detecting one type of action instead of detecting a broad spectrum of different actions.

Much like the development process of action recognition, extreme action recognition requires an AI that can interpret patterns in the temporal and spatial domains. Hence, the complexity of the problem increases, and the use of a time-series network becomes a necessity.

The current standard approach to a recognition framework typically uses one or a combination of feature extractors such as 2D CNN, optical flow, background subtraction, pose estimation, and ConvLSTM, to extract the salient spatio-temporal features from an image. A classifier would then be used on the extracted feature to make a prediction. Examples of the classifier are SVM, 3D CNN, or FC layer. This sub-chapter explores the existing extreme action recognition framework developed by researchers.

2.5.1 Recognition with SVM as classifier

As of now, the SOTA AI classifiers are CNN and the RNN. However, the SVM remains a popular choice as a classifier due to their low computational cost and high speed. The work by Bilinski et al. (2018) explored the possibility of building a violent behaviour detection AI without using any neural network for each video, the histogram of oriented gradients, histogram of optical flow, trajectory shape, and motion boundary histogram in both X-Y directions were used to capture spatial features. The extracted spatial features were then processed with Improved Fisher Vector to obtain a temporal representation of the sequence of the frames.

By conducting the classification using SVM, the authors achieved 93.7% accuracy on the Hockey dataset, 99.5% accuracy on the Movies dataset, and 96.4% accuracy on the Violent-Flows dataset. In Bilinski's work, a detection framework using sliding windows was also implemented to locate the boundaries of the fight. However, in the dataset, no bounding box ground truth was provided. Hence, the Intersection of the Union of detection cannot be evaluated. (Bilinski et al., 2018)

As shown by Bilinski et al. (2018) approach, extreme action recognition can be performed without reliance on neural networks. Therefore, a baseline for the usage of neural networks is set. However, the use of SVM meant that parameters of SVM is limited to a particular dataset and must be tuned to each specific dataset.

In a different work by Carneiro et al. (2019), multi-feature extraction and multi-stream 2D CNN were combined as the feature extractors and SVM were used as the classifier. The multiple features extracted from a video were optical flow, depth, visual rhythm and red, green, blue (RGB) images. The four features were each connected to an independent stream of VGG-16 2D CNN, and as a result, four distinctly different features could be learnt by CNN without conflict.

The VGG-16 was modified that the last two dense layers are trainable individual stream learners, and the network weights were initialized by transfer learning with weights trained from ImageNet. The output of the final dense layer from each stream was concatenated into an ensemble. The ensemble served as the input of SVM, which performed the classification. The described architecture is illustrated in Figure 2.18, showing the type of extracted features of optical flow, depth, visual rhythm, and RGB from counting top to bottom, respectively. (Carneiro et al., 2019)

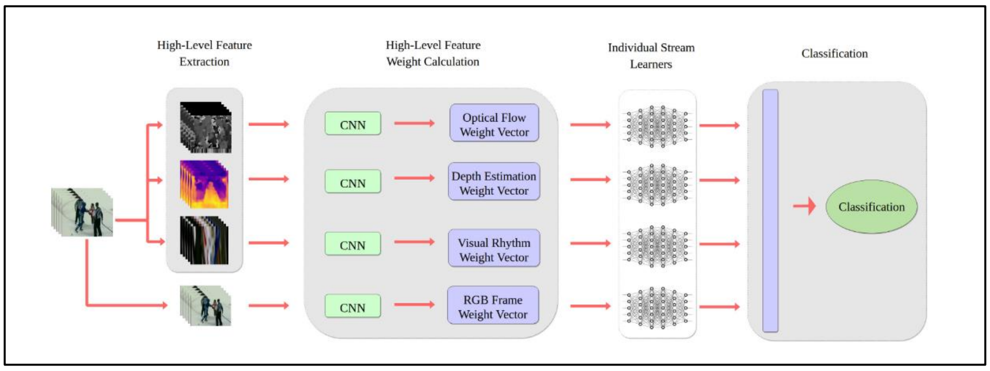
High-Level Feature Extraction
High-Level Feature Weight Calculation
Individual Stream Learners
Classification
CNN
Optical Flow Weight Vector
CNN
Depth Estimation Weight Vector
CNN
Visual Rhythm Weight Vector
CNN
RGB Frame Weight Vector
Classification

Figure 2.18: Model Architecture of Multi-Stream Learning. (Carneiro et al., 2019)

According to the author, 88.62% on the Hockey dataset and 100% accuracy on the Movies dataset were reported. If the visual rhythm stream were removed, the accuracy of the Hockey dataset would improve to 89.10%. Nevertheless, the work by Carneiro et al. (2019) had limitations, such as the model was slow and computationally intensive. As a result of using four independent VGG-16 streams and one non-trainable depth estimation CNN in the framework, it is the most computationally intensive framework presented in this sub-chapter. In addition, although VGG-16, which has a relatively lower number of layers, was selected by the author for his work, upon inspection, the overall computation cost is actually much higher than other CNN models available to the author.

For example, to perform one forward pass, VGG-16 need to perform 7,800 Million Multiply-Accumulations (MMACCs) operations, whereas MobileNet V2 published in 2018 only has to perform 569 MMACC, both of the networks have a similar top-5 error of 10%. The training process also did not utilise regularisation; thus, the trained model may tend towards overfitting.

A different work by Nova, Ferreira and Cortez (2019) attempted to use human pose information as an input feature for the recognition framework. The OpenPose CNN was used to estimate human poses in an image. However, OpenPose is incapable of establishing multi-person pose estimation. Therefore, the pose tracking technique was performed using kernelised correlation filters to draw each person's region of interest (ROI) before performing the pose estimation. Once the pose estimation was performed, human pose information such as body joints velocities and body shape boundary were derived. Examples of this human pose information are demonstrated in Figure 2.19. The human pose information was used as an input for the SVM for learning and recognition output. The author reported a true positive accuracy of 85% and true negative accuracy of 92% when tested on the ISR-UOL 3D Social Activity dataset. This work demonstrated that identifying human poses may function well as an extreme action recognition input feature as the movements of joints is a potential indication of a fight. However, because the ISR-UOL 3D dataset is not a notable benchmark for extreme action recognition, a comparison cannot be drawn between this work and other works. (Nova, Ferreira and Cortez, 2019)

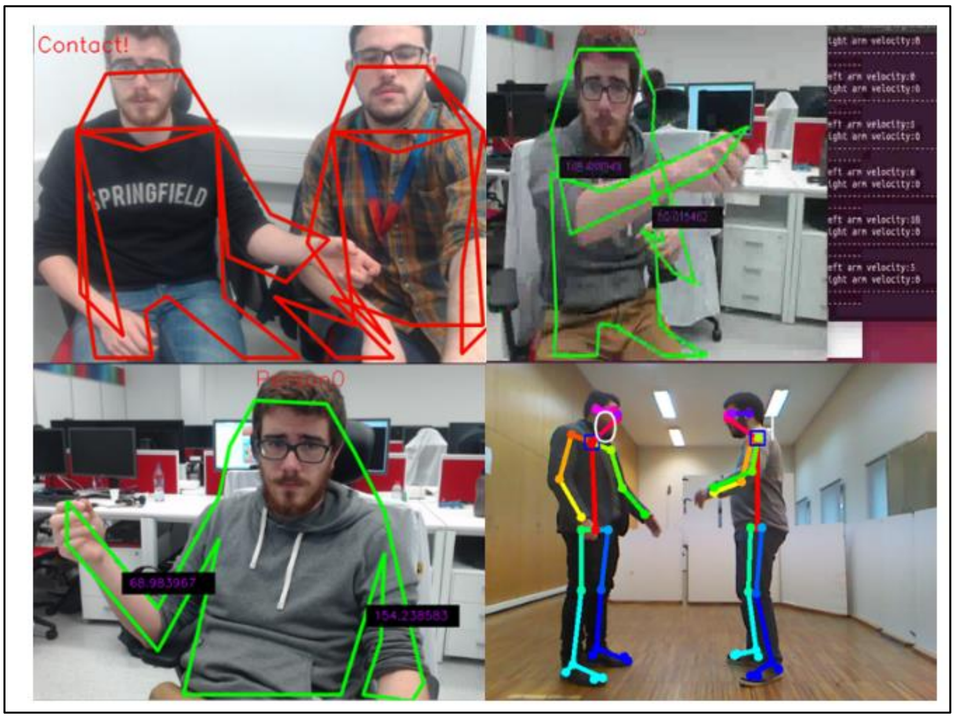

Figure 2.19: Example of human pose information extracted in the study with OpenPose. (Nova, Ferreira and Cortez, 2019)

2.5.2 Recognition with 3D CNN as classifier

As an extension of the established 2D CNN, 3D CNN is currently the most popular option in training an AI for extreme action recognition framework, as it is implemented in most deep learning python libraries such as Tensorflow, Caffe, and Pytorch with complete documentation of use.

Ullah et al. (2019) proposed a framework using a combination of Single Shot Detector (SSD) and 3D CNN. First, the SSD was used to draw bounding boxes on an area of the image where there were people, and the bounding boxes served as a region of interest to perform classification. Then, the bounding box area was extracted and sent as an input to 3D CNN for recognition. The architecture described can be seen in Figure 2.20. Using a bounding box for recognition helps reduce noise by removing part of the background and allowing the 3D CNN to learn only on the area where there is a person. The best performance reported by the author achieved an accuracy of 96% on the hockey dataset, 99.9% on the movies dataset, and 98% on the violent crowd dataset.

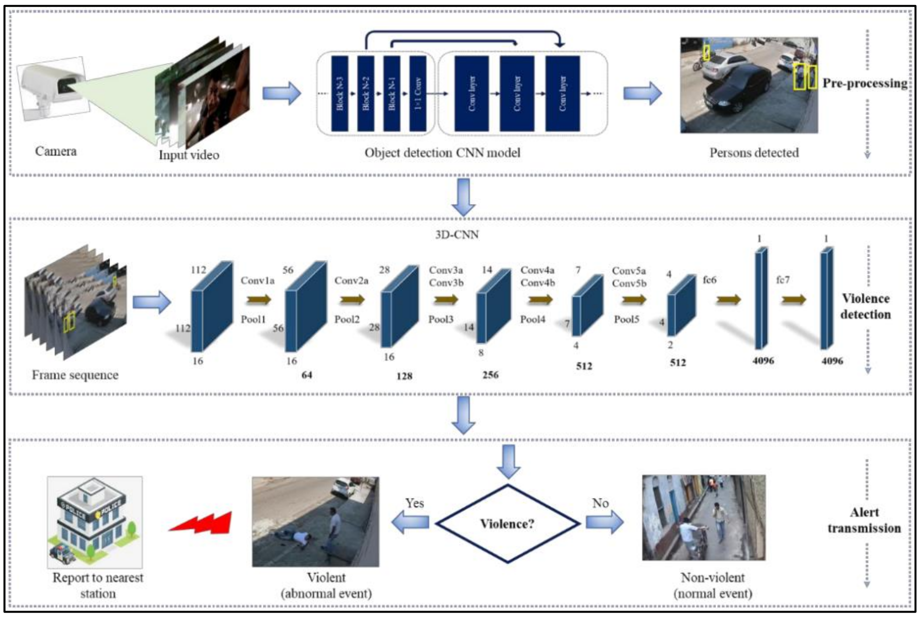

Figure 2.20: Model Architecture of method SSD + 3D CNN. (Ullah et al., 2019)

One notable observation can be made from the data provided by the author; although the author has trained the model for up to 5000 epochs, test results from training were recorded once every 500 epochs. As shown in the trained model inFigure 2.21, there has been no performance improvement since the first test result as it plateaued between 1000 to 5000 epochs. In order to prevent overfitting, an early stopping mechanism can be implemented together with an increase in test frequency.

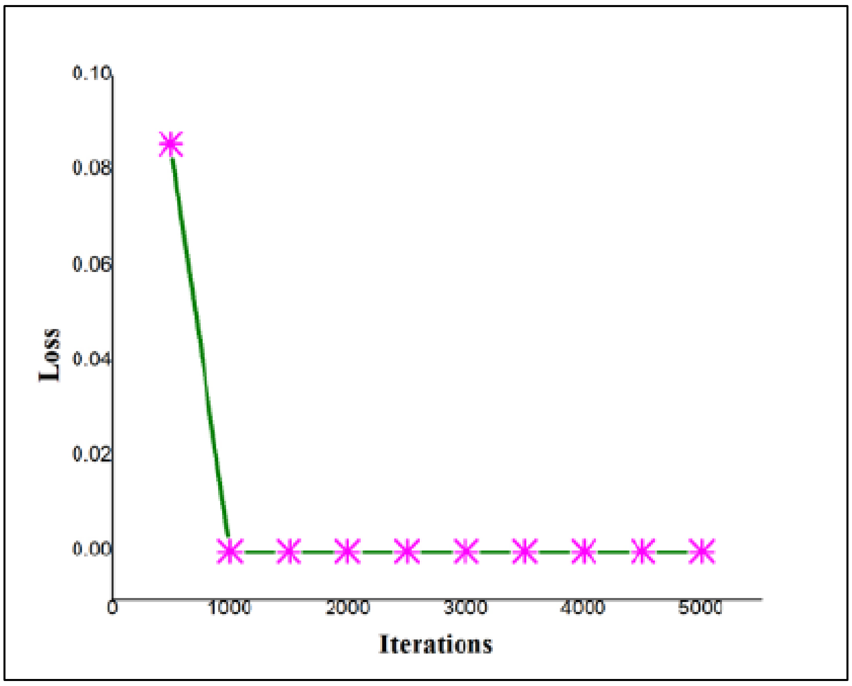

Figure 2.21:Graph of epochs plotted against loss, showing long period of no improvements after 1000 epochs. (Ullah et al., 2019)

A different work by Xu, See, and Lin (2019) proposed using SSD in combination with an optical flow motion activation map to more accurately localise the region of interest for extreme action recognition. The methodology implemented by the author in Figure 2.22 was to separate the framework into two branches: the localisation branch and the recognition branch.

In the localisation branch, SSD was used to detect and draw bounding boxes of each person, together with the optical motion activation map to draw bounding boxes on the image area where motion was intense. The multiple bounding boxes output from SSD and optical motion activation map were merged into several ROI using non-max suppression (NMS) with custom alignment criterion; as a result, each ROI included a cluster of few persons. (Xu, See and Lin, 2019)

In the recognition branch, a two-stream 3D CNN takes the input of the unprocessed RGB frame, motion acceleration map, and the ROI from the localisation branch to compute the recognition output. This method obtained 98.6% accuracy on Hockey and 99.8% on Movies. (Xu, See and Lin, 2019)

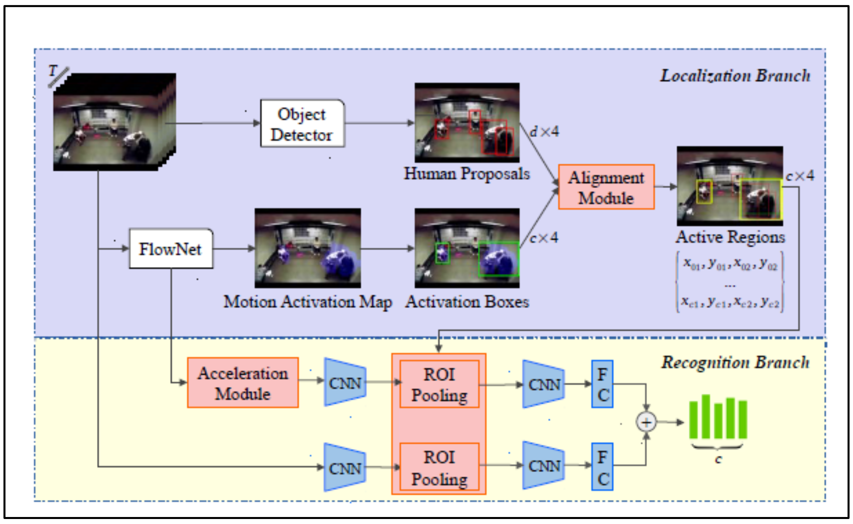

Figure 2.22:Model Architecture of method SSD combined with Optical Flow and 3D CNN, the recognition framework is separated into localisation branch and recognition branch. (Xu, See and Lin, 2019)

Both methodology by Xu, See, and Lin (2019) and Ullah et al. (2019) included extracting a local region of interest as input for 3D CNN. In contrast, Li et al. (2019) showed that it is possible to train an end-to-end EAR framework using only 3D CNN, without the need for other explicit regions of interest, and the input was RGB image frames from the video. The model architecture shown in Figure 2.23 demonstrated that no pre-processing was required. The author tested their work on three datasets and obtained 98.3% accuracy on the Hockey dataset, 100% accuracy on the Movies dataset, and 97.17% on the Violent-Flows dataset.

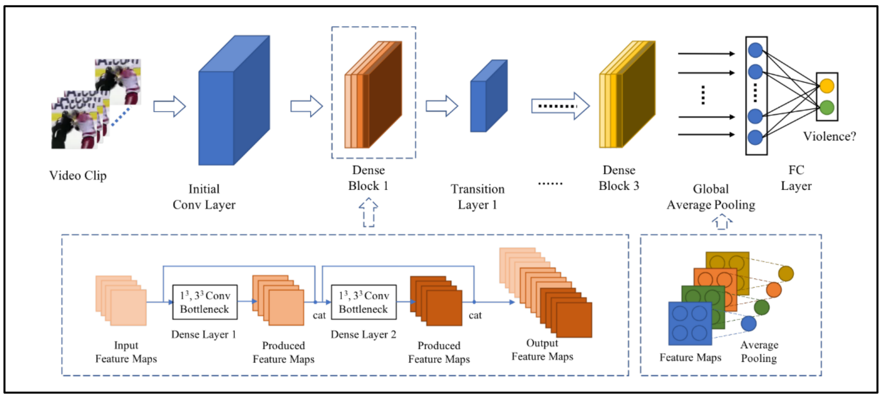

Figure 2.23:The model architecture of end-to-end 3D CNN capable of recognition without pre-processing steps. (Li et al., 2019)

Calzavara (2020) extended on the work of Li et al. (2019) by adding a DensePose-RCNN pose estimation pre-processing step to the 3D CNN developed by Li et al. (2019). Illustrated in Figure 2.24, the model architecture of DensePose-RCNN developed by the author would draw a mask of human keypoints over every person detected within the frame. Subsequently, the 3D CNN would learn and make a prediction on the output from DensePose-RCNN instead of using RGB input frames directly. Calzavara (2020) reported that the addition of the pose estimation pre-processing step yielded an accuracy of 96.7% on hockey, 100% on movies, 97.2% on Violent-Flows.

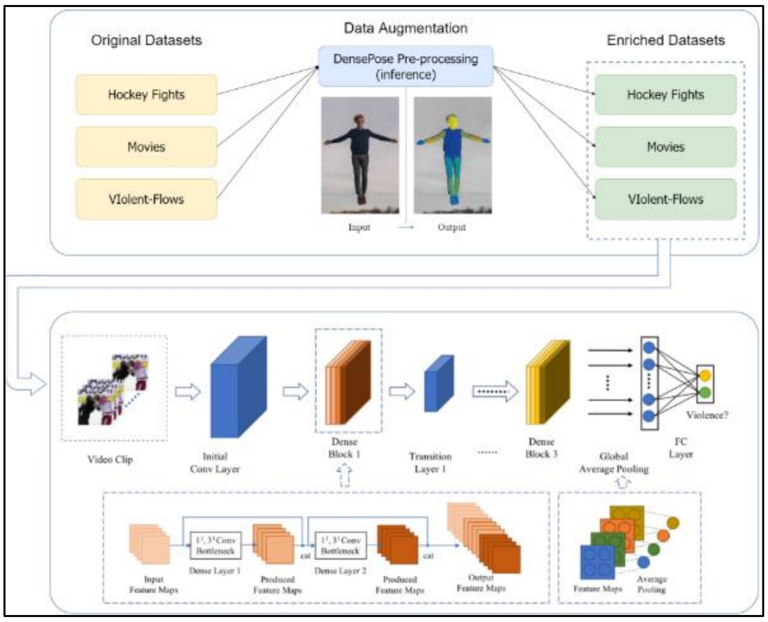

Figure 2.24: Model architecture of densePose RCNN combined with 3D CNN. (Calzavara, 2020)

The pose estimation pre-processing step is implemented on the premise that the movements of human limbs and torso are critical factors in deciding whether a person is fighting. However, based on the work done by Calzavara (2020), the addition of the pre-processing step does not significantly impact the accuracy, most likely due to the errors introduced by the use of DensePose-RCNN. Nevertheless, a more accurate pose estimation AI may improve results.

2.5.3 Recognition with ConvLSTM as classifier

LSTM, like other RNNs, has only been used in applications where input is a 1D vector such as translation and speech recognition. However, using LSTM in computer vision has been made possible with the introduction of ConvLSTM by Shi, Chen, Wang, and Yeung (2015). Although the effectiveness of ConvLSTM in extreme action recognition remains a topic to be researched, compared to many well-established methods such as SVM and 3D CNN, there are fewer research articles that use ConvLSTM as the classifier.

An article published by Sudhakaran and Lanz (2017) utilised ConvLSTM to build an extreme action recognition framework. The model architecture illustrated in Figure 2.25 shows that the image was first pre-processed to extract motion information by performing background subtraction between image frames. AlexNet 2D CNN without the classification layer was then used to extract spatial features. Next, the extracted features were sent to ConvLSTM to encode spatial features into spatial-temporal features. Lastly, the FC layer was connected to the last memory cell of ConvLSTM to make a prediction.

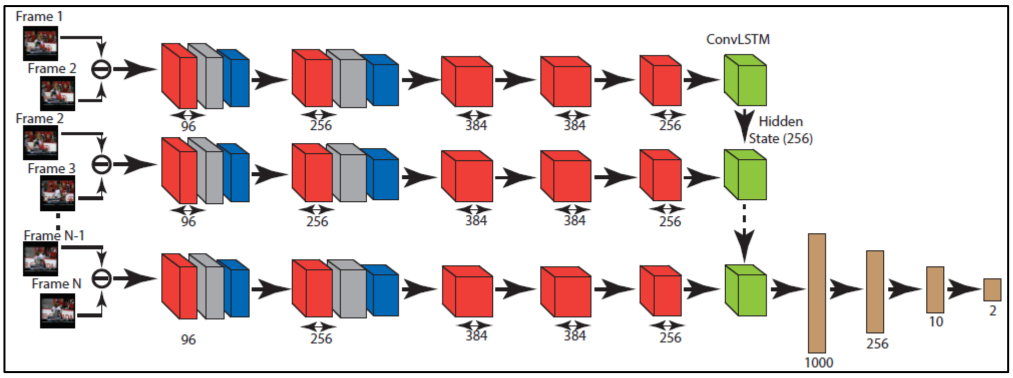

Figure 2.25: The model architecture of 2D CNN combined with ConvLSTM. (Sudhakaran and Lanz, 2017)

Sudhakaran and Lanz (2017) reported an accuracy of 97.1% on the Hockey dataset, 100% on the Movies dataset, and 94.57% on the Violent-Flows dataset. However, there was a limitation on the author’s work; even though AlexNet was relatively advanced at the time of publication, it is considered obsolete as a feature extractor by current standards. There are more effective and efficient 2D CNN being published in more recent works. Changing the

AlexNet to a newer 2D CNN such as MobileNet V3 may improve the result substantially.

A SOTA was achieved by Islam et al. (2021) using two-stream separable ConvLSTM. In the first stream, the model architecture illustrated in Figure 2.26 shows that RGB image frame with background suppression was taken as input, MobileNetV2 was used to extract spatial features, and lastly, ConvLSTM was used to encode spatio-temporal information. The second stream was similar, except that the input was pre-processed with frame difference. The outputs of the separable ConvLSTM from both streams were then concatenated into an ensemble, and the prediction was made using the FC regression layer on the ensemble.

Moreover, Islam et al. (2021) demonstrated the effectiveness of ConvLSTM with Hockey Dataset at 99.50% accuracy and the Movies dataset at 100% accuracy. Notably, 89.75% was obtained for the RWF-2000 dataset, the primary benchmark with one of the highest number of videos dedicated for extreme action recognition.

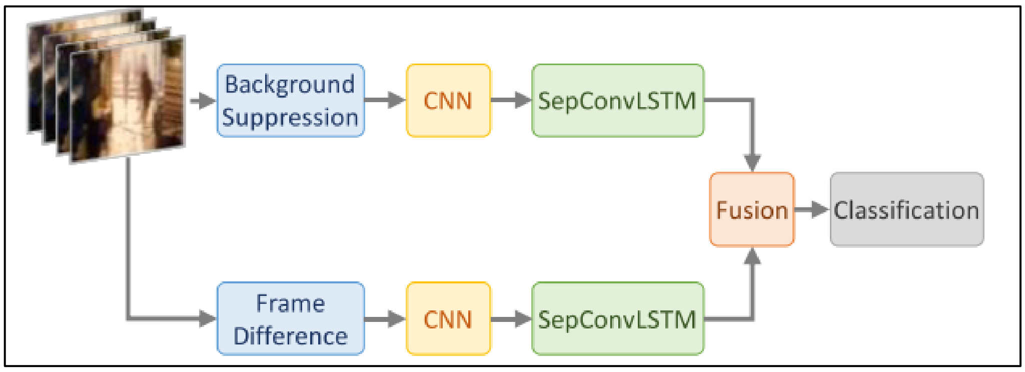

Figure 2.26: Model architecture of dual-stream 2D CNN + Separable ConvLSTM. (Islam et al., 2021)

2.5.4 Motivation

The results of the literature reviewed in sub-chapter 2.5.1 to sub-chapter 2.5.3 can be summarized in Table 2.2. The (-) symbol indicates that the method did not perform any test with that specific dataset.

Table 2.2: Comparison of Accuracy for Reviewed Methodologies.

Method	**Dataset Benchmark (% accuracy)**			
	Hockey	**Movies**	**VF**	**RWF**
IFV + SVM Bilinski *et al.* (2018)	93.70	99.50	96.40	-
Multistream 2D CNN + SVM Carneiro *et. al.* (2019)	88.62	100.00	89.10	-
OpenPose + SVM Nova, Ferreira and Cortez (2019)	-	-	-	-
SSD + 3D CNN Ullah *et al.* (2019)	96.00	99.90	98.00	-
SSD + Optical Flow + Two-stream 3D CNN Xu, See and Lin (2019)	98.60	99.80	-	-
3D CNN Li *et al.* (2019)	98.30	100.00	97.17	-
DensePose + 3D CNN Calzavara (2020)	96.70	100.00	97.20	-
2D CNN + ConvLSTM Sudhakaran and Lanz (2017)	97.10	100.00	94.57	-
Two-stream 2D CNN + ConvLSTM Islam *et al.* (2021)	99.50	100.00	-	89.75

As of date, there is no existing research that specifically used pose information as the input for 2D CNN spatial feature extraction + ConvLSTM spatio-temporal encoder. In both articles featuring ConvLSTM, the techniques used for pre-processing were background suppression or frame differences. This study believes that developing a framework that uses pose information as an input feature will be worthwhile. The hypothesis is that background suppression, and background subtraction would impart background movements from non-human moving objects; objects such as cars on roads and highways will show up as giant blobs of motion, leading to false-positive detection. In comparison, by utilising the pose information of the human body, the trained model will be relatively unperturbed by noises, leading to an increase in recognition performance.

Although this hypothesis is not founded on a theoretical basis, there is evidence suggesting that using pose information as the input can improve performance. In work by Calzavara (2020), a pose information pre-processing step was added to the model developed by Li et al. (2019), and despite the addition of an inaccurate pose estimation AI to generate pose information to use as input, Calzavara (2020) has managed to achieve near-identical performance. This study believes that should a more accurate SOTA pose estimation model be used, the accuracy may well surpass the model that did not implement pose information.

This study is motivated to use pose information as the input for 2D CNN feature extraction based on the hypothesis. The choice of using 2D CNN as the feature extraction is supported by the results obtained in the work by Sudhakaran and Lanz (2017). In his work, high accuracy can be achieved even with the obsolete AlexNet, and this study believes that 2D CNN has a strong ability to extract spatial information. Subsequently, in work by Islam et al. (2021), ConvLSTM outperforms other spatio-temporal encoding methods. Hence, in this study, the ConvLSTM network is chosen.

Lastly, Islam et al. (2021) and Ullah et al. (2019) have shown that the complexity of Hockey, Movies and VF datasets are not sufficient as the error approaches Bayes error. Therefore, the emerging dataset of RWF-2000 should be used in addition to the three datasets, as the RWF-2000 dataset allows model evaluation at a larger scale.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In Figure 3.1, the initial plan to develop the EAR framework is shown. The framework consists of an N-1 number of time-series networks, serially connected at the spatio-temporal encoder. Each time-series network takes input from two images that are adjacent to each other as a sequence. Thus, for example, the first network input is the sequence of images one and two, and the following network input is the sequence of images two and three. The network will then propagate through the entire length of the videos.

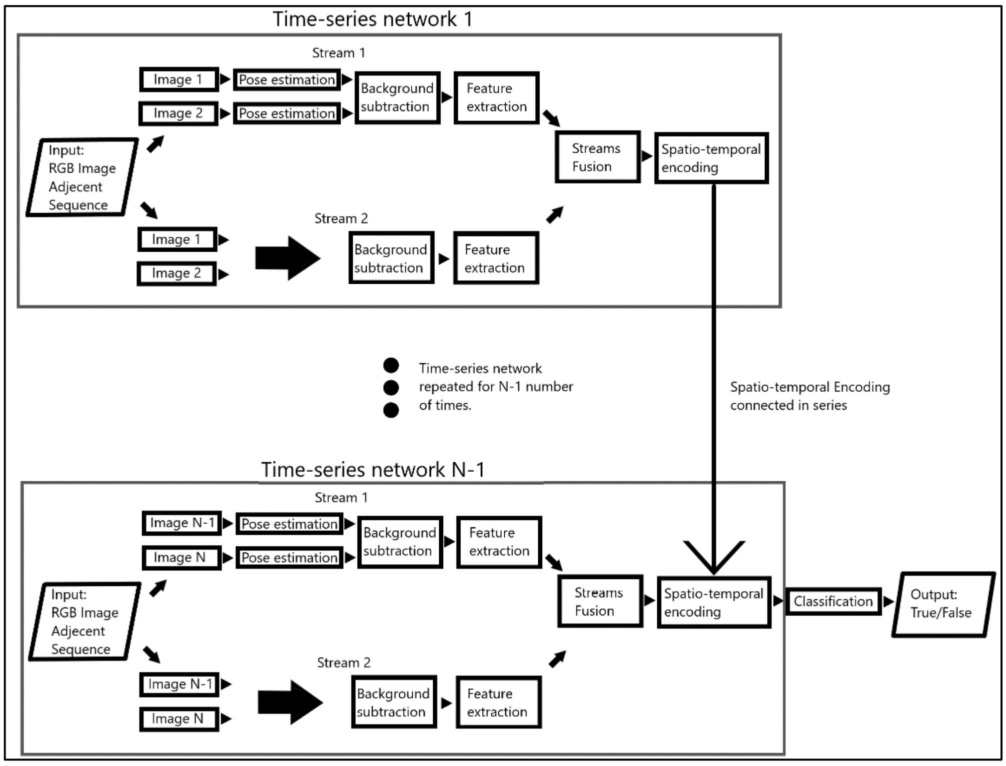

Figure 3.1: Initial plan to develop the extreme action recognition framework, each box illustrates the process involved.

Each time-series network is split into two streams, and the first stream pre-processes its input of two images by pose estimation to extract pose keypoints images. Background subtraction is then performed on the pose keypoints images to emphasise motion between pose keypoints images.

Subsequently, a trainable 2D CNN will be used to extract features from the subtracted pose keypoint image. The first stream is planned based on this study's hypothesis, which believes that including pose information as part of the input will improve recognition accuracy. Furthermore, pose estimation is necessary as existing benchmarks only provided RGB images for both training and testing datasets.

The second stream is identical to the first, except pose estimation and background subtraction is directly performed on the RGB input images. The purpose of the second stream is to complement the model in situations where pose estimation did not perform as desired. The two-stream setup has been shown to work extraordinarily well in work by Islam et al. (2021), and in this study, a two-stream setup is used to examine the effectiveness of pose information in the extreme action recognition.

The outputs of the 2D CNNs from both streams will be concatenated to form an ensemble for spatio-temporal encoding. Then, the spatio-temporal encoders are serially connected to form a linked memory network, and the spatio-temporal encoder of the final network will be connected to the classification layer that outputs a true/false label.

3.2 Dataset to be used for framework training

Based on the findings from sub-chapter 2.1.6, this study has selected the Hockey, Movies, VF, and RWF-2000 datasets for training and testing. The training and testing will be conducted on each dataset separately, which allow the model's performance to be compared to existing literature. In addition, the train-test split will be randomised to 80/20 split for Hockey, Movies, and VF datasets, whereas RWF-2000 dataset has a fixed 80/20 split pre-determined by the author of the dataset.

3.3 Proposed framework

3.3.1 Input of the framework

Each video in the datasets is typically a few seconds long, and depending on the dataset, each video may contain between 90 to 270 frames. In order to reduce computational costs, not all the frames are utilised in this study. An even sample of 20 images from each video is extracted as a sequence to represent the overall motion in the video. The equation for even sampling is shown in Equation (3.1

$$i = round\left((n-1) \times \frac{L}{N-1}\right) \qquad (3.1)$$

where

n = sequence of {1,…,N}

L = Total number of frames in video

N = sequence length

i = index of frame to be sampled

3.3.2 Pose estimation

Pose information is extracted from an RGB image using a pose estimation 2D CNN model in the pre-processing step. As discussed in the literature review, it is crucial to select a pose estimation model as accurately as possible to minimise the amount of error introduced into the input.

Furthermore, it is essential to select an efficient model. For example, if a sequence length is 20 images, then for each sequence, the pose estimation model needs to perform 20 inferences on each image. Hence, it is essential to choose an efficient model that will reduce computational costs. This study has identified EfficientPose as the most suitable pose estimation to be used. This selection is justified in Table 3.1. By comparing EfficientPose to other pose estimation models available to date, EfficientPose can achieve excellent mean Average Precision (mAP) and mean Average Recall (mAR) with low computational cost. Moreover, EfficientPose outperforms DensePose, chosen by Calzavara (2020) and OpenPose, chosen by Nova, Ferreira and Cortez (2019).

Table 3.1: Table of available options for pose estimation model tested on the COCO test-dev dataset. (Güler, Neverova and Kokkinos, 2018; Cao et al., 2021; Zhang et al., 2021)

Method	**Backbone**	**GFLOPs**	**mAP**	**mAR**
Mask-RCNN	ResNet-50-FPN	-	63.1	-
G-RMI	ResNet-101	57.0	64.9	69.7
CFN	-	-	72.6	-
RMPE	PyraNet	26.7	72.3	-
CPN	ResNet-Inception	-	72.1	78.5
SimpleBaseLine	ResNet-50	8.9	70.0	75.6
SimpleBaseLine	ResNet-152	15.7	71.6	77.3
HRNet-W32	HRNet-W32	16.0	74.9	80.1
HRNet-W48	HRNet-W48	32.9	75.5	80.5
LPN	ResNet-50	1.0	68.7	74.5
LPN	ResNet-101	1.4	70.0	75.7
LPN	ResNet-152	1.8	70.4	76.2
PNFS	MobileNet-V2	4.0	67.4	73.1
PNFS	ResNet-50	11.4	70.9	76.6
DensePose + Mask	-	-	52.8	62.0
OpenPose	-	-	61.8	-
EfficientPose A	NAS searched	0.5	66.5	-
EfficientPose B	NAS searched	1.1	70.5	76.1
Efficient Pose C	NAS searched	1.6	70.9	76.5

Multi-person pose estimation is essential in this study as extreme actions typically involve more than one person. However, EfficientPose cannot be directly deployed to be used for multi-person pose estimation in an image. In order to overcome this limitation, this study refers to the open-source HRNet single-person pose estimation.

HRNet was successful at implementing multi-person pose estimation using a single-person model. In their work, Faster R-CNN was first used to draw bounding boxes on every person within an image, followed by performing single-person pose estimation in every bounding box. The methodology implemented by HRNet is replicated in this study, except for choice for person detection; the SOTA object detection framework YOLOv5 is used instead of Faster R-CNN. As of date, the journal publication on YOLOv5 has not been completed; however, it is the improved version of YOLOv4 with higher frames per second (FPS) and mAP, and YOLOv4 has been shown to outperform Faster-RCNN and other available options shown in Table 3.2. (Wang et al., 2020)

Table 3.2: Tables of object detection framework showing available options for open-source object detection frameworks (Bochkovskiy, Wang and Liao, 2020)

Method	**Backbone**	**Size**	**FPS**	**mAP**
EfficientDet-D0	Efficient-B0	512	62.5	33.8
EfficientDet-D1	Efficient-B1	640	50.0	39.6
EfficientDet-D2	Efficient-B2	768	41.7	43.0
YOLOv3 + ASFF*	Darknet-53	320	60	38.1
YOLOv3 + ASFF*	Darknet-53	416	54	40.6
YOLOv3 + ASFF*	Darknet-53	608	45.5	42.4
RFBNet	HarDNet68	512	41.5	33.9
RFBNet	HarDNet85	512	37.1	36.8
Faster-RCNN	ResNet-50	-	9.4	39.8
YOLOv4 (P)	CSPDarknet-53	416	54	41.2
YOLOv4 (P)	CSPDarknet-53	512	43	43.0
YOLOv4 (P)	CSPDarknet-53	608	33	43.5
YOLOv4	CSPDarknet-53	416	96	41.2
YOLOv4	CSPDarknet-53	512	83	43.0
YOLOv4	CSPDarknet-53	608	62	43.5

The output of the pose estimation is in the form of a keypoints coordinate list. This study uses the keypoints coordinate list to draw the joints over a black canvas without background or contextual information, as shown in Figure 3.2. The removal of background aims to minimise the effect of background noise. As a result, this study theorises that the pose information stream will thoroughly learn from the pose, and any contextual information will be learned in the RGB stream.

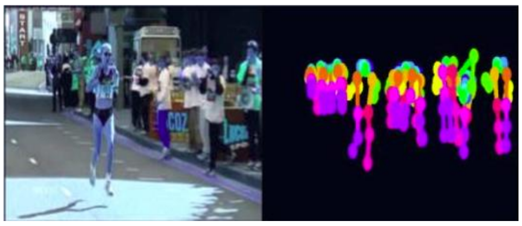

Figure 3.2: The image shown on the left is the original video frame before pose estimation; the image on the right shows pose keypoints of the pedestrians drawn on the black canvas after pose estimation.

3.3.3 Background subtraction

The frame differencing approach is chosen for background subtraction in this study to capture motion information in the form of change in pixels. In addition, frame differencing does not require a specific background to be defined and only requires two images to function. In contrast, other approaches such as mean and median filters require more than two images to establish a running average, which puts additional demand on computational resources. The Frame difference technique assumes that the background of the current frame is the same as the previous frame and computes the difference by taking the absolute value of subtraction between the current frame and the previous frame. Frame differences are mathematically computed between adjacent frames at the pixel level with Equation (3.2. (Tamersoy, 2009)

$$B_{x,y,z,n} = \left|P_{x,y,z,n} - P_{x,y,z,n-1}\right| \tag{3.2}$$

where

B = background subtracted pixel

P = pixel value of frame

x = index of image width

y = index of image height

z = index of colour channel

n = index of image sequence

3.3.4 2D CNN feature extractor

In a framework, a feature extractor is an essential part that extracts salient features from the image. In this study, the selected feature extractor has to perform up to 20 inferences for each prediction. For this reason, the use of an efficient network will reduce overall inference time and training time to an acceptable length of time.

As shown in Table 3.3, MobileNet V2 and V3 can achieve performance comparable to the previous SOTA of VGG 16 or GoogleNet with less computational cost. This study selects Mobilenet V3, as it has lower latency compared to MobileNet V2. Specifically, MobiletNet V3 large is selected for the pose information stream, and MobileNet V3 small for the RGB stream to emphasise pose information while reducing the training and inference time. (PyTorch, 2021; Howard et al., 2019; Véstias, 2019)

Table 3.3: Table of open-source options of 2D CNN that can be used for feature extraction. (Véstias, 2019; Howard et al., 2019; PyTorch, 2021)

Method	**MMACCs**	**Latency (ms)**	**Top-5 Accuracy (%)**	**Top-1 Accuracy (%)**
AlexNet	650	-	79.07	56.52
VGG 16	7800	-	90.38	71.59
ResNet 101	3800	-	93.55	77.37
ResNet 152	5650	-	94.05	78.31
Inception V3	5700	-	93.45	77.29
GoogleNet	750	-	89.53	69.78
DenseNet 201	1500	-	93.37	76.90
MobileNet V2	300	162	90.29	71.88
MobileNet V3 Small	-	43	87.40	67.67
MobileNet V3 Large	-	119	91.34	74.04

3.3.5 ConvLSTM spatio-temporal encoder

The equations of the ConvLSTM used in this study are identical to those described in sub-chapter 2.4.2, incorporated with convolution principles described in sub-chapter 2.4.3. The convolution aspect is evident in the forget gate equation in Equation 3.3, the gate component at Equation 3.4, the memory cell component at Equation 3.5, and the output gate at Equation 3.6, wherein the weight (w), input (I), and hidden state (h) convolve, instead of direct multiplication as used in standard LSTM. (Shi et al., 2015)

$$f_t = \sigma\left(w_x^f * I_t + w_h^f * h_{t-1} + b^f\right) \tag{3.3}$$

$$i_t = \sigma\left(w_x^i * I_t + w_h^i * h_{t-1} + b^i\right) \tag{3.4}$$

$$\tilde{c}_t = tanh\left(w_x^{\tilde{c}} * I_t + w_h^{\tilde{c}} * h_{t-1} + b^{\tilde{c}}\right) \tag{3.5}$$

$$o_t = \sigma(w_x^o * I_t + w_h^o * h_{t-1} + b^o) \tag{3.6}$$

$$c_t = \tilde{c}_t \odot i_t + c_{t-1} \odot f_t \tag{3.7}$$

$$h_t = o_t \odot \tanh\,(c_t) \tag{3.8}$$

where

f = forget gate

i = gate component

$\tilde{c}$ = memory cell component

o = output gate

c = cell state

h = hidden state

t = time steps

I = image input vector

w = weight vector

b = bias vector

σ = sigmoid function

$tanh$ = hyperbolic tangent function

$*$ = convolution function

The direct multiplication of Equation (3.7 and Equation (3.8 likewise is replaced with the Hadamard product to accommodate the multiplication of

2D arrays. The ConvLSTM equations are better illustrated as a whole in Figure 3.3; similar to standard LSTM shown in Figure 2.16, each memory cell can be connected serially at the cell and hidden state.

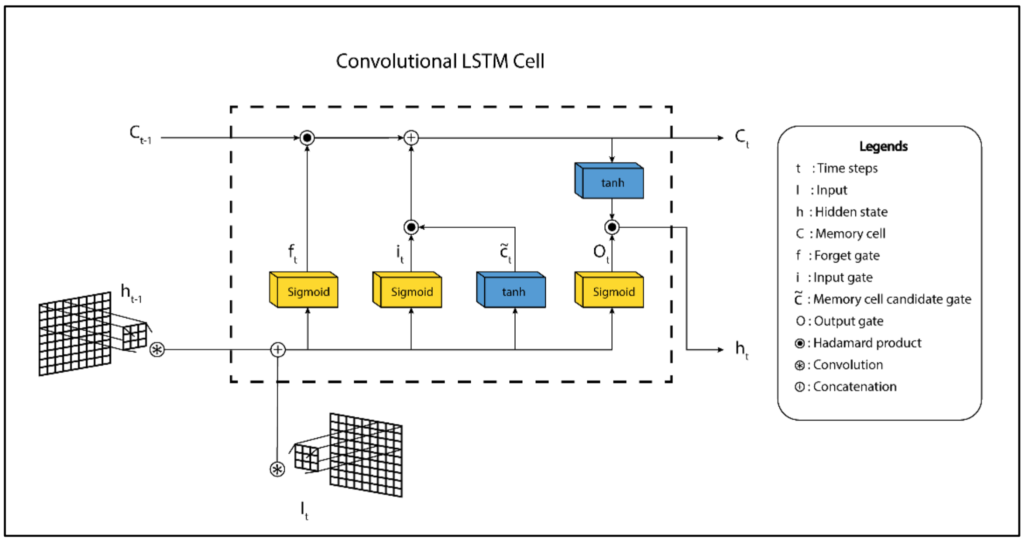

Figure 3.3: Graphical model of a ConvLSTM memory cell.

3.3.6 Classification

The architecture of the linear layer is shown in Table 3.4, where the hidden state of the final ConvLSTM cell is connected to the first layer of the classification layers. The first layer max pool the hidden state from a 3D array of size (1536,7,7) to (1536,3,3) with a kernel size of (2,2) and stride of two, and automatically flatten the 3D array to 1D vector from (1536,3,3) to (13824). Layers 2 to 8 learn predict the occurrence of extreme action from the flattened hidden state.

Table 3.4: Architecture of Classification layers for the proposed framework.

Layer Number	**Layer type**	**Input**	**Output**
1	Maxpool	(1536,7,7)	13824
2	Linear	13824	1536
3	Batch normalisation	1536	1536
4	ReLU	1536	1536
5	Linear	1536	512
6	ReLU	512	512

Table 3.4 (Continued)

7	Linear	512	10
8	ReLU	10	2
9	Linear	10	2

3.3.7 Overall framework

Figure 3.4 summarises all components described from sub-chapter 3.3.1 to 3.3.6 and is consistent with the initial plan shown in Figure 3.1. The multi-person pose estimation stream will be processed using a combination of YOLOv5 with EfficientPose to produce keypoints frames, and the weights of both pose estimations will be frozen and set to untrainable. Frame difference background subtraction is computed for keypoints frames and RGB frames; subsequently, feature extraction is performed on the frame difference using MobileNet V3 large for keypoints frame, and MobileNet V3 small for RGB frames. The extracted features are concatenated and spatio-temporally encoded with ConvLSTM through time. The networks are serially stacked at the point of ConvLSTM hidden state and cell states. Hidden state output from the final ConvLSTM is sent to the classification layers for prediction. The trainable components in the network include the MobileNet V3 feature extractors, ConvLSTM, and linear layers in the classification layers.

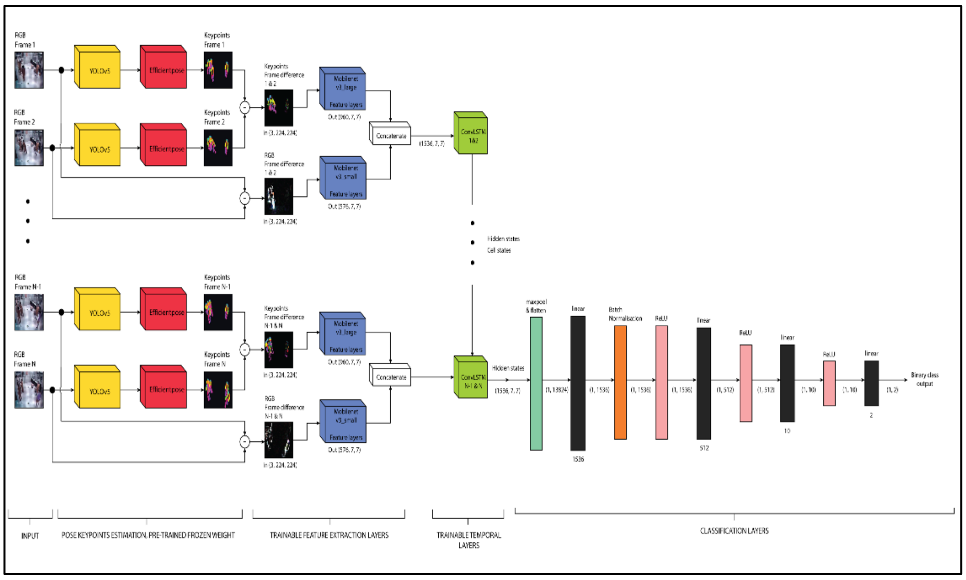

Figure 3.4: The proposed framework of this project showing all significant components.

3.4 Experiment and ablation study

The experiment will be set up to train the model with 100 epochs, and test results will be collected at every epoch. The low number of epochs for training in this study is decided based on the occurrence of overfitting in work by Ullah et al. (2019), evidenced by a long period of no improvement in Figure 2.21; a higher number of epochs can only be justified if loss continues to be minimised at 100 epochs.

Quantitative results of train/test accuracy, false-positive rate, and false-negative rate will be collected at every epoch for analysis to evaluate the performance of the proposed framework. Epoch numbers at the point of no improvement will be collected to evaluate the model complexity and identify the occurrence of overfitting. Pose keypoints frames and RGB frame differences for failure cases will be recorded for qualitative analysis on potential causes of failure.

An ablation study will be conducted to identify the overall effectiveness of the framework by training and testing the framework with either one of the streams being disabled. Each stream is expected to have the predictive capability on its own, but the predictive capability of using each stream alone should be lower than using both streams combined.

3.5 Development environment

The specifications of the deep learning hardware used in this study are listed in Table 3.5. The cloud computing solution was considered but not chosen; the comparison between two popular options for cloud computing is shown in Table 3.6. Colab is free of charge; however, the graphics processing unit (GPU) availability is limited, and the runtime limit is capped at 12 hours, whereas Colab Pro guarantee GPU availability with a runtime limit increased to 24 hours. Having a runtime limit of 24 hours is risky to the project because deep learning training can easily exceed 24 hours. Other better cloud computing options such as Microsoft Azure and Amazon Web Service has a minimum rental period which is not suitable for this study. (Google Colab, 2021)

Table 3.5: Specification of hardware for deep learning.

Component	Model
CPU	AMD Ryzen 3700X
GPU	Nvidia RTX3080 10GB VRAM
RAM	64GB DDR4 3200MHz
Memory	2TB read/write speed: 3.5/3.0GHz
OS	Ubuntu 20.04

Table 3.6: Two popular options of cloud computing solutions showing the estimated monthly cost, the GPU availability, and the runtime limit. (Google Colab, 2021)

Specification	**Type of Cloud Computing Solution**	
	Colab Free	**Colab Pro**
Cost/month	Free	RM 41.26
GPU availability	Limited use Nvidia K80	Nvidia P100
Runtime limit	12 hours	24 hours

The framework is written entirely in Python 3.8 on the PyCharm community version. Among other popular options such as Matlab or C++, Python was chosen for its extensiveness of diversity in modules. Pythons allow industry-leading deep learning modules such as TensorFlow, Caffe, or Pytorch to be imported and used as a library to accelerate framework development. In addition, multi-threaded backpropagation can be computed using functions provided by the deep learning modules, which reduces training time.

The deep learning library chosen in this study is PyTorch 1.8.0 for its computational graph checkpoint function. This function overcomes the limitation of having low video random-access memory (VRAM) in the RTX3080. RTX3080 is a GPU designed for the gaming market with a limited 10GB VRAM, and training CNN + ConvLSTM with only 10GB VRAM is not practical without reducing batch size. By default, all intermediate activations computed during forward-pass are stored in the VRAM for backpropagation. The checkpoint function computes intermediate activations on-demand in backpropagation by tracking weights and inputs during forward-pass. The on-demand computation lowers the training speed, which, in turn, allows large models to be trained with limited VRAM.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Quantitative Results

4.1.1 Accuracy

Table 4.1 shows the accuracy obtained with the framework proposed in this study with different variations in methods. Unfortunately, the proposed method of RGB + EfficientPose C did not outperform the models developed in the existing literature. However, it did obtain results comparable to the model developed by Sudhakaran and Lanz (2017) regarding the Hockey, Movies, and VF datasets. In addition, the proposed model was able to perform with 2.75% lower accuracy than the current SOTA developed by Islam et al. (2021) in the most complex dataset of RWF-2000.

Table 4.1: Accuracy obtained with proposed framework and ablation study compared to major benchmarks from previous studies.

	Dataset Benchmark (% accuracy)			
Method	**Hockey**	**Movies**	**VF**	**RWF**
IFV + SVM Bilinski *et al.* (2018)	93.70	99.50	96.40	-
Multistream 2D CNN + SVM Carneiro *et. al.* (2019)	88.62	100.00	89.10	-
SSD + 3D CNN Ullah *et al.* (2019)	96.00	99.90	98.00	-
SSD + Optical Flow + Two-stream 3D CNN Xu, See and Lin (2019)	98.60	99.80	-	-
3D CNN; Li *et al.* (2019)	98.30	100.00	97.17	-
DensePose + 3D CNN Calzavara (2020)	96.70	100.00	97.20	-

Table 4.2 (Continued)

2D CNN + ConvLSTM Sudhakaran and Lanz (2017)	97.10	100.00	94.57	-
Two-stream 2D CNN + ConvLSTM Islam *et al.* (2021)	99.50	100.00	-	89.75
Method conducted in this study	**Hockey**	**Movies**	**VF**	**RWF**
RGB + EfficientPose A	96.00	100.00	94.00	86.25
RGB + EfficientPose B	96.50	100.00	94.00	86.25
RGB + EfficientPose C (Proposed)	97.00	100.00	92.00	87.00

A trend is observed with different variations of EfficientPose. The increase of pose estimation accuracy tends to improve framework performance on Hockey and RWF datasets, and diminishing return is observed on the Violent-Flow dataset. This observation suggests that the pose estimation model was overwhelmed in crowded scenarios.

Table 4.3: Result obtained from ablation study.

Method	**Dataset Benchmark (% accuracy)**			
	Hockey	**Movies**	**VF**	**RWF**
RGB only	96.00	100.00	94.00	85.00
EfficientPose C only	93.50	100.00	88.00	83.50

4.1.2 Training analysis

Figure 4.1 aggregates the training characteristics of the proposed method, RGB+EfficientPose C, for all datasets. One distinct observation made on the graphs of train and test loss, except for the Movies dataset, is that the model overfitted within the first two epochs when trained with Hockey, Violent-Flows, and RWF-2000 datasets. However, despite overfitting, the model still exhibited excellent accuracy performance. The most probable cause was exploding gradients; the gradients computed from the backpropagation were too big. The large gradients caused a rapid increase in accuracy and decrease in loss, and the large gradients eventually caused the loss to overshoot and oscillate around a local minimum point. The effects of exploding gradients can be minimised using gradient clipping techniques and decreasing the value of weights initialisation.

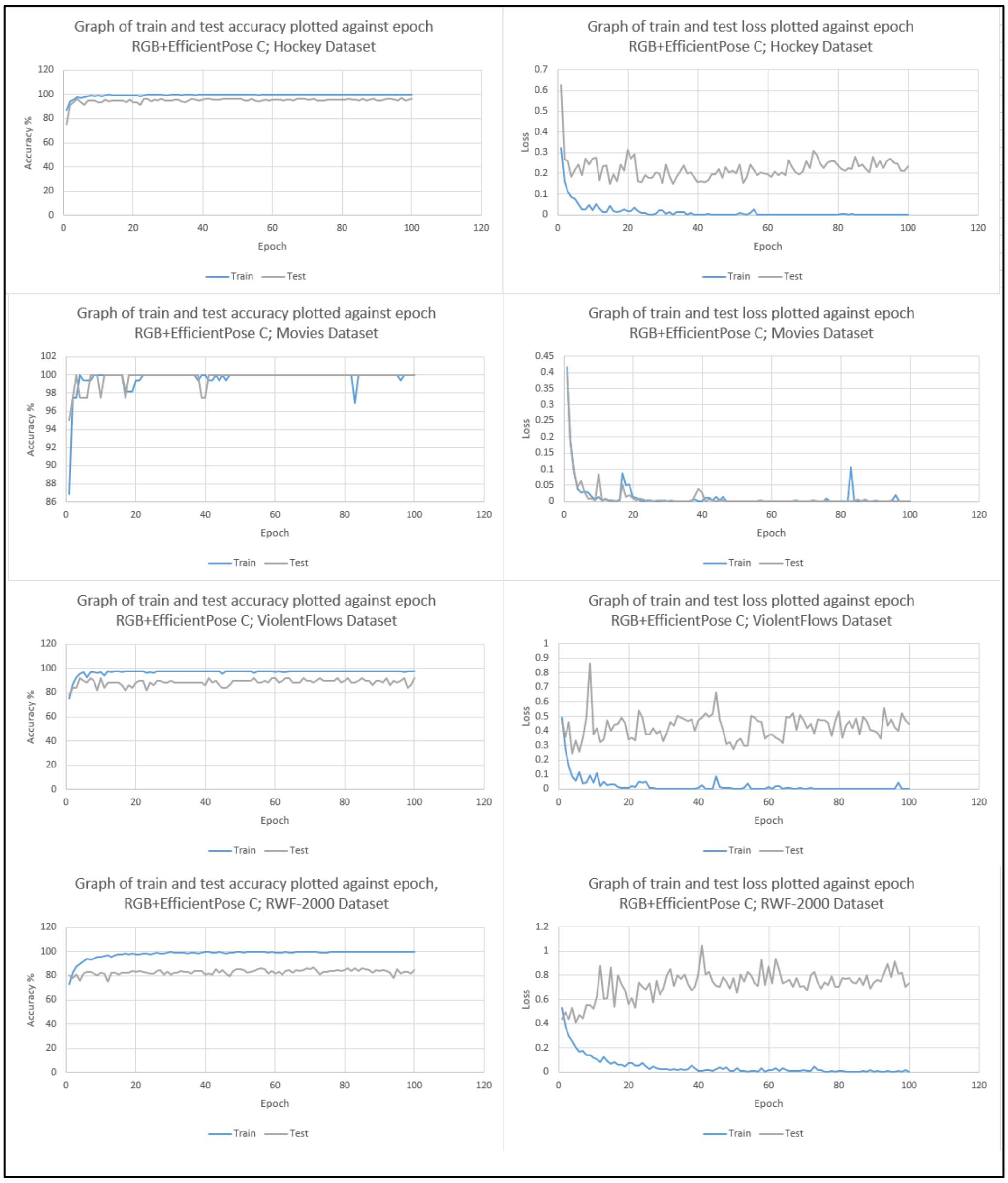

Figure 4.1: Graph of loss and accuracy for method of RGB + EfficientPose C on all datasets during training, plotted against epoch.

4.1.3 Training time

Figure 4.2 shows the training time for different methods, and a disadvantage is seen for the proposed method. The stream without pose estimation required 40 seconds to train per epoch, whereas the stream using pose estimation required 529 seconds. Likewise, the proposed method of RGB+EfficientPose C utilised pose estimation and required 591 seconds to train, resulting in approximately

1377% more time to train. Training time scales linearly with the size of the dataset. Hence, datasets larger than RWF-2000 will require an even longer training time per epoch.

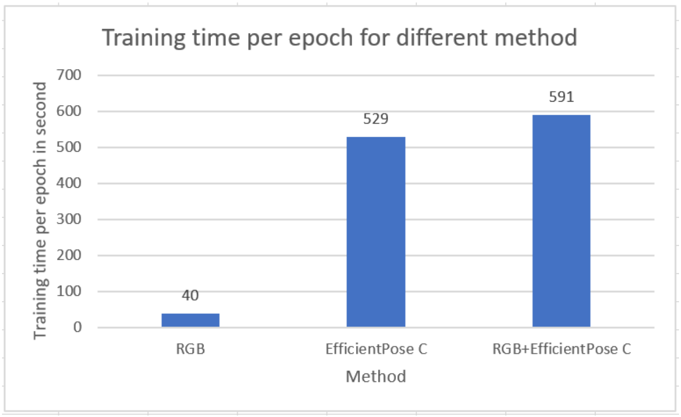

Figure 4.2: Training time taken per epoch for different method training on RWF-2000 dataset.

4.2 Qualitative Results

The qualitative analysis conducted in this sub-chapter will focus on failure analysis on the proposed method of RGB+EfficientPose C. Videos that the framework failed to predict will be analysed to identify root causes of failure in the framework.

Figure 4.3 next page depicts eight frames extracted from the Hockey dataset that were incorrectly predicted as a fighting scene. In the unprocessed image, the hockey player maintained an upright posture; however, the pose estimation process failed to estimate correctly and represented the hockey player with exaggerated motions shown at the bottom eight frames. Thus, the false positive was most likely caused by incorrect poses estimation by analysing the poses keypoints.

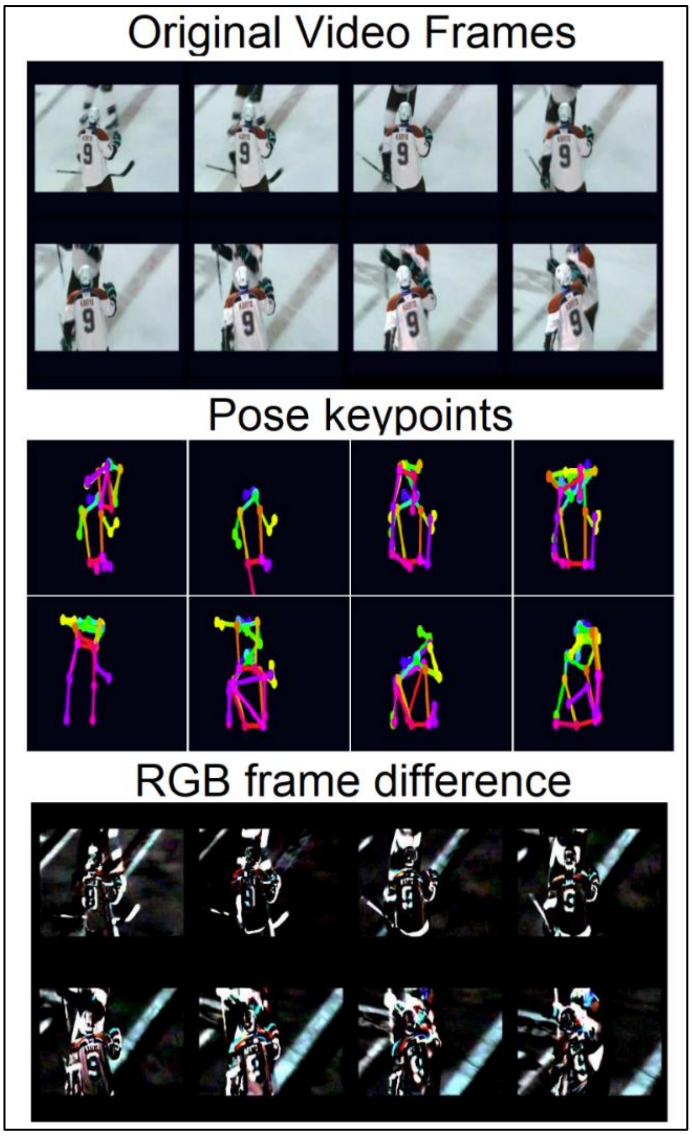

Figure 4.3: False positive case with method RGB+EfficientPose C with sample from hockey dataset showing incorrect pose estimation

The video clips in Figure 4.4 next page depict an argument in a café with multiple people entering and exiting the scene; although there were movements in the video, no fights were occurring. Thus, both the pose estimation and RGB frame difference appear to function as intended. However, by analysing part of the RGB frame difference in detail, as shown in Figure 4.5, dispersed red colour static noises can be observed. Thus, the movement of the people in the scene combined with the erratic appearances of the noises most likely caused the framework to make a false-positive prediction.

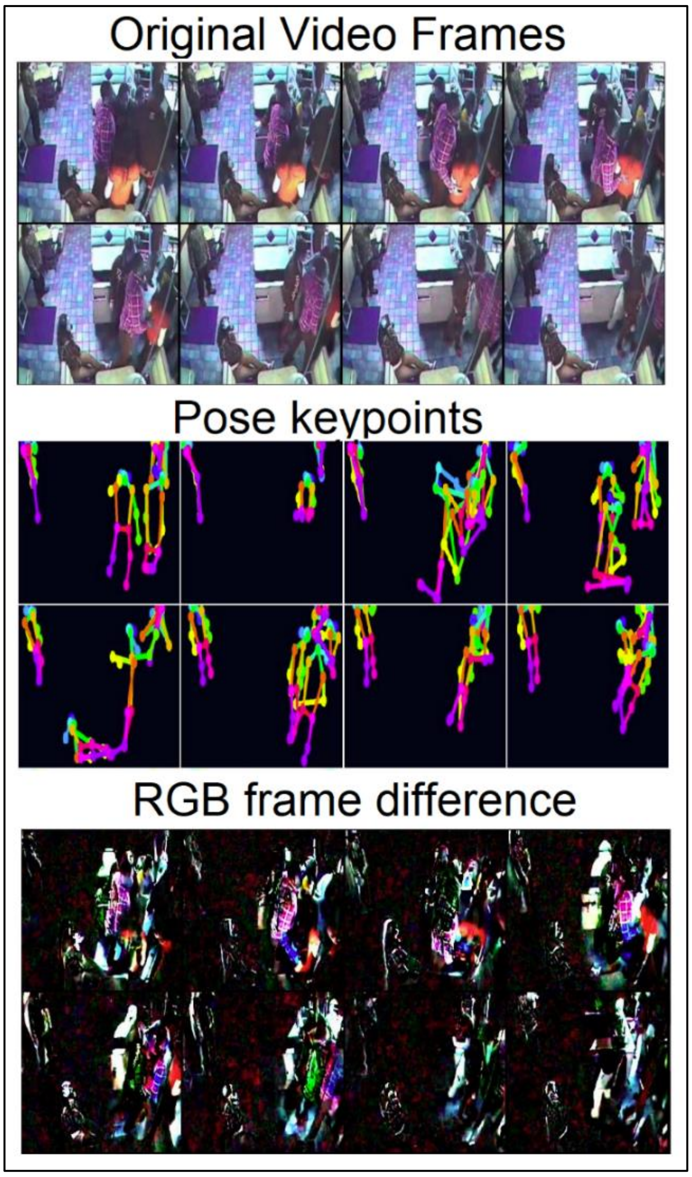

Figure 4.4: False positive case with method RGB+EfficientPose C with sample from RWF-2000. The pose estimation correctly predicted the pose keypoints in most cases.

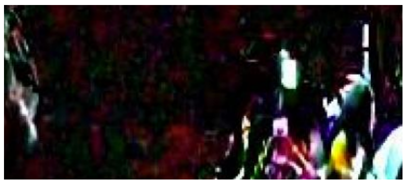

Figure 4.5: Dispersed red colour static noises observed from the RGB frame difference in the false positive case with method RGB+EfficientPose C with sample from RWF-2000.

Figure 4.6 shows the scene of some pedestrians walking in a car park before cutting to a different scene inside a building. The pose keypoints and RGB frame difference both appear to function well. However, the scene change caused an apparent shift in poses, and the final frame in frame difference became overwhelmed. Thus, the scene change during time-series inference may have been falsely predicted to be extreme action.

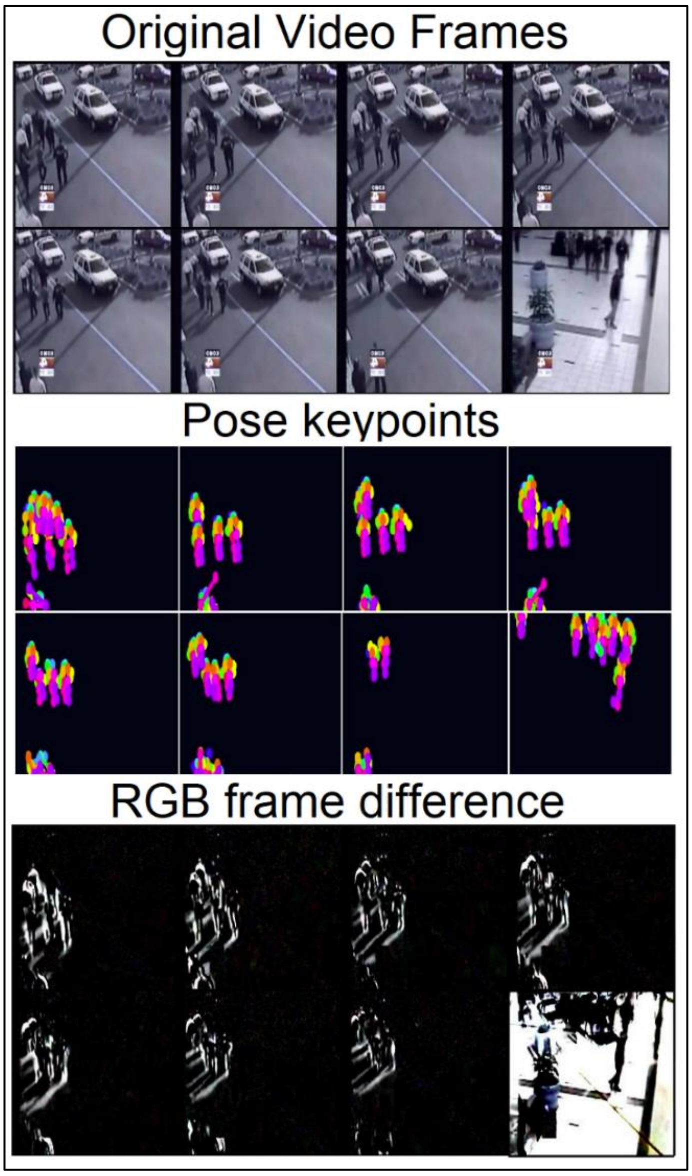

Figure 4.6: False positive case with method RGB+EfficientPose C with test sample from RWF-2000, drastic scene change observed at the final frame.

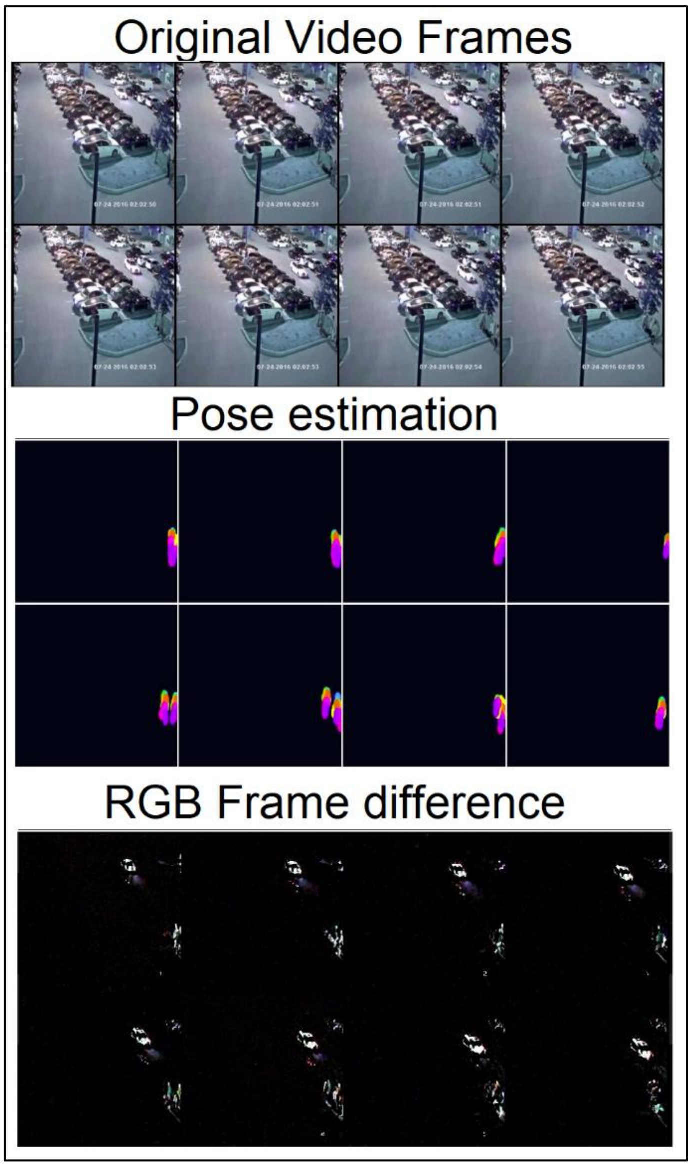

Figure 4.7: False negative case with method RGB+EfficientPose C with sample from RWF-2000 dataset depicting correct pose estimation, but subjects of interest were too small.

Figure 4.7 depicts a false negative case in a video from RWF-2000. A fight occurred at the bottom right corner of the video, and pose estimations were computed with remarkable accuracy despite the size of the persons in the image. However, the framework fails to recognise the fight, most likely due to camera placement being too far away and the subjects of interest were too small for the framework to extract useful motion information. Similarly, the frame difference stream cannot discern any indication of motions that could suggest extreme action. A region of interest pre-processing algorithm may help to improve accuracy by isolating the subjects of interest.

Figure 4.8 shows a false negative case from Violent-Flows. Crowd fights were occurring in the scene; however, the pose estimation failed to capture keypoints of people in the frame due to the poor sharpness of the image. The RGB frame difference stream similarly failed to capture any meaningful motions due to the motion of the camera and noise of the image.

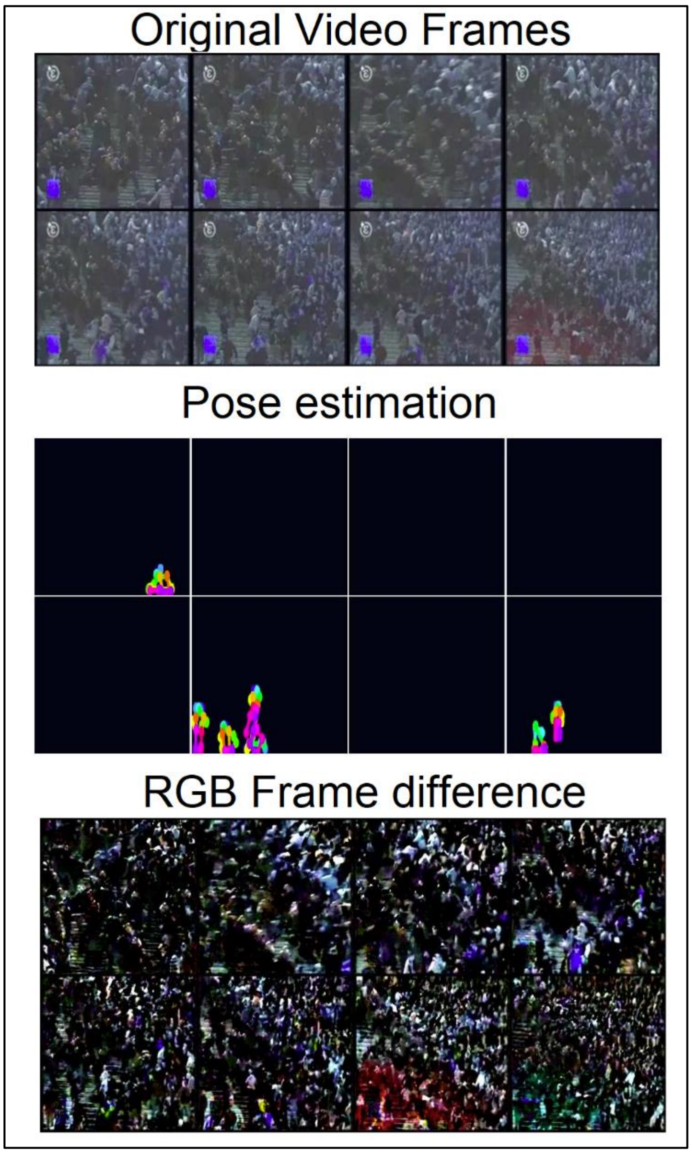

Figure 4.8: False negative case with method RGB+EfficientPose C with sample from Violent-Flows crowd dataset depicting incorrect pose estimation and excessive noise in RGB frame difference.

4.3 Summary of critical findings

The results obtained in this project revealed several critical findings. Firstly, complementing the RGB frame difference with pose keypoints will improve the framework's accuracy in most cases, as shown in the ablation study. However, the pose estimation model increased computation cost drastically despite the improvement, evidenced by the 1377% training and testing time increase.

Training analysis displayed indications of exploding gradient in framework during training with the datasets of Hockey, Violent-Flows, and RWF-2000. Therefore, gradient clipping should be added to prevent exploding gradients, and the learning rate should also be tuned with a smaller initialisation weight.

Furthermore, qualitative analysis shows the framework appears to be resistant against background noises such as the motion of cars, as no instances of such failure cases can be observed. However, the framework remains susceptible to static noise, sudden scene changes, small subjects, and blurry images. Lastly, the framework has been shown to consistently fail when pose estimation cannot accurately predict the person's pose keypoints in the image.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusions

In conclusion, this project proposed an extreme action recognition framework with dual-stream MobileNet V3 CNN integrated with ConvLSTM. The development of the proposed framework was motivated by the need to overcome the existing limitation identified in literature reviews. The proposed framework was trained on standard datasets of Hockey, Movies, Violent-Flows, and RWF-2000.

The proposed framework obtained comparable performance to existing SOTA on the RWF-2000 dataset at 87.00% accuracy, 100% accuracy on the Movie dataset, 97.00% accuracy on the Hockey dataset; however, the proposed method performed poorly on the Violent-Flows dataset at 92% accuracy. Ablation study shows that dual-stream RGB frame difference with pose keypoints will improve the framework's accuracy at the cost of 1377% increase in training and testing time, and training analysis revealed indications of exploding gradient for framework trained and tested with Hockey, Violent-Flows, and RWF-2000 datasets.

Lastly, qualitative analysis shows that the proposed framework displays exceptional noise resistance against background motion consistent with the hypothesis of this study; however, the framework remains susceptible to static noises, scene change, small subjects, blurry image, and failure of pose estimation.

5.2 Recommended Solutions

In future work, semantic segmentation can be considered an alternative to pose estimation to filter background noises. Currently, the computational resources required to use pose estimation may prove challenging for most practical applications, such as home surveillance using edge devices. In addition, pose estimation have been observed in qualitative analysis to make frequent mistakes. Although semantic segmentation cannot discriminate between pose keypoints of each person, it can provide a comparatively more accurate human body contour at a reduced computational cost than pose estimation.

Furthermore, the learning rate and weight initialisation should be tuned for each dataset and the implementation of gradient clipping to reduce the exploding gradient. Moreover, adding a region proposal algorithm will allow the framework to focus on the region of interest that will reduce framework susceptibility to small subjects. Finally, the effects due to blurry images and static noises may be reduced by adding denoising algorithm pre-processing.

REFERENCES

Adilah, A., 2017. 40,000 CCTV cameras to make FT smart and safe. *Malay Mail*. [online] 15 Dec. Available at: <https://www.malaymail.com/news/malaysia/2017/12/15/40000-cctv-cameras-to-make-ft-smart-and-safe/1532911>.

Bilinski, P., Bremond, F., Bilinski, P., Bremond, F., Violence, H., Videos, S., Bilinski, P., Bremond, F. and Antipolis, I.S., 2018. Human Violence Recognition and Detection in Surveillance Videos To cite this version : HAL Id : hal-01849284 Human Violence Recognition and Detection in Surveillance Videos.

Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y.M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. [online] Available at: <http://arxiv.org/abs/2004.10934>.

Calzavara, I., 2020. Human pose augmentation for facilitating Violence Detection in videos : a combination of the deep learning Ivan Calzavara.

Cao, Z., Hidalgo, G., Simon, T., Wei, S.E. and Sheikh, Y., 2021. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1), pp.172–186.

Carneiro, S.A., Da Silva, G.P., Guimaraes, S.J.F. and Pedrini, H., 2019. Fight detection in video sequences based on multi-stream convolutional neural networks. *Proceedings - 32nd Conference on Graphics, Patterns and Images, SIBGRAPI 2019*, pp.8–15.

Cheng, M., Cai, K. and Li, M., 2019. RWF-2000: An Open Large Scale Video Database for Violence Detection. [online] Available at: <http://arxiv.org/abs/1911.05913>.

Chilamkurthy, S., 2017. *TRANSFER LEARNING FOR COMPUTER VISION TUTORIAL*. [online] PyTorch. Available at: <https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html> [Accessed 16 Apr. 2021].

Chollet, F., 2020. *Transfer learning & fine-tuning*. [online] Available at: <https://keras.io/guides/transfer_learning/> [Accessed 16 Apr. 2021].

Freudenrich, C. and Boyd, R., 2021. *How Your Brain Works*. [online] Available at: <https://science.howstuffworks.com/life/inside-the-mind/human-brain/brain1.htm> [Accessed 16 Apr. 2021].

Github, 2021. *Looking inside neural nets*. [online] Available at: <https://ml4a.github.io/ml4a/looking_inside_neural_nets/> [Accessed 16 Apr. 2021].

Google Colab, 2021. *Choose the Colab plan that's right for you*. [online] Available at: <https://colab.research.google.com/signup> [Accessed 6 Sep. 2021].

Gracia, I.S., Suarez, O.D., Garcia, G.B. and Kim, T.K., 2015. Fast fight detection. *PLoS ONE*, 10(4), pp.1–19.

Güler, R.A., Neverova, N. and Kokkinos, I., 2018. DensePose: Dense Human Pose Estimation in the Wild. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.7297–7306.

Hassner, T., Itcher, Y. and Kliper-Gross, O., 2012. Violent flows: Real-time detection of violent crowd behavior. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. [online] Providence, RI, USA: IEEE.pp.1–6. Available at: <https://ieeexplore.ieee.org/document/6239348>.

Hochreiter, S. and Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Computation*, [online] 9(8), pp.1735–1780. Available at: <https://direct.mit.edu/neco/article/9/8/1735-1780/6109>.

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Le, Q. and Adam, H., 2019. Searching for mobileNetV3. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October, pp.1314–1324.

IBM Cloud Education, 2020. *Convolutional Neural Networks*. [online] Available at: <https://www.ibm.com/cloud/learn/convolutional-neural-networks> [Accessed 16 Apr. 2021].

Islam, Z., Rukonuzzaman, M., Ahmed, R., Kabir, M.H. and Farazi, M., 2021. Efficient Two-Stream Network for Violence Detection Using Separable Convolutional LSTM. [online] Available at: <http://arxiv.org/abs/2102.10590>.

Ji, S., Xu, W., Yang, M. and Yu, K., 2013. 3D Convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1), pp.221–231.

Li, J., Jiang, X., Sun, T. and Xu, K., 2019. Efficient violence detection using 3D convolutional neural networks. *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2019*, (October 2020).

Mahidin, M.U., 2020. *CRIME STATISTICS, MALAYSIA, 2020*. [online] Available at: <https://www.dosm.gov.my/v1/index.php?r=column/cthemeByCat&cat=455&bul_id=UFZxVnpONEJqUU5pckJIbzlXeEJ1UT09&menu_id=U3VPMldoYUxzVzFaYmNkWXZteGduZz09>.

Matlab, 2021. *Convolutional Neural Network*. [online] Available at: <https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html> [Accessed 16 Apr. 2021].

Nova, D., Ferreira, A. and Cortez, P., 2019. A Machine Learning Approach to Detect Violent Behaviour from Video BT - Intelligent Technologies for Interactive Entertainment. pp.85–94.

Oquab, M., Bottou, L., Laptev, I. and Sivic, J., 2014. Learning and transferring mid-level image representations using convolutional neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.1717–1724.

PyTorch, 2021. *TORCHVISION.MODELS*. [online] Available at: <https://pytorch.org/vision/stable/models.html> [Accessed 6 Sep. 2021].

Rosebrock, A., 2019. *Deep Learning for Computer Vision with Python*. 3rd ed. Pyimagesearch.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W. and Woo, W., 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *Journal of Sensors*, [online] 2018, pp.1–9. Available at: <http://arxiv.org/abs/1506.04214>.

Soh, M.B.C., 2012. Crime and Urbanization: Revisited Malaysian Case. *Procedia - Social and Behavioral Sciences*, 42(July 2010), pp.291–299.

Soomro, K., Zamir, A.R. and Shah, M., 2012. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. [online] (November). Available at: <http://arxiv.org/abs/1212.0402>.

Sudhakaran, S. and Lanz, O., 2017. Learning to detect violent videos using convolutional long short-term memory. *arXiv*.

Suzuki, K., 2011. *ARTIFICIAL NEURAL NETWORKS- ARCHITECTURES AN APPLICATIONS Edited by Kenji Suzuki*.

Tamersoy, B., 2009. *Background Subtraction*. [online] Available at: <https://www.cs.utexas.edu/~grauman/courses/fall2009/slides/lecture9_background.pdf> [Accessed 6 Sep. 2021].

Ullah, F.U.M., Ullah, A., Muhammad, K., Haq, I.U. and Baik, S.W., 2019. Violence detection using spatiotemporal features with 3D convolutional neural network. *Sensors (Switzerland)*, 19(11), pp.1–15.

Véstias, M.P., 2019. A survey of convolutional neural networks on edge with reconfigurable computing. *Algorithms*, 12(8).

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W. and Xiao, B., 2020. Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (March), pp.1–1.

Wayne, 2020. *Scratch to SOTA: Build Famous Classification Nets 2 (AlexNet/VGG)*. [online] Medium. Available at: <https://medium.com/swlh/scratch-to-sota-build-famous-classification-nets-2-alexnet-vgg-50a4f55f7f56> [Accessed 16 Apr. 2021].

Wood, T., 2021. *Convolutional Neural Network*. [online] DeepAI. Available at: <https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network> [Accessed 16 Apr. 2021].

Xu, Q., See, J. and Lin, W., 2019. Localization guided fight action detection in surveillance videos. *Proceedings - IEEE International Conference on Multimedia and Expo,* 2019-July, pp.568–573.

Yu, Y., Si, X., Hu, C. and Zhang, J., 2019. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, [online] 31(7), pp.1235–1270. Available at: <http://arxiv.org/abs/1803.01446>.

Zhang, W., Fang, J., Wang, X. and Liu, W., 2021. EfficientPose: Efficient human pose estimation with neural architecture search. *Computational Visual Media*, 7(3), pp.335–347.