AUTOMATED DETECTION OF VIOLENT CONTENTS FOR FILM CENSORSHIP USING DEEP LEARNING

by

MUHAMMAD SHAHRIL NIZAM BIN ABDULLAH

1171102891

Session 2022/2023

The project report is prepared for

Faculty of Engineering

Multimedia University

in partial fulfilment for

Bachelor of Engineering (Hons) Electronics

majoring in Telecommunications

FACULTY OF ENGINEERING

MULTIMEDIA UNIVERSITY

February 2023

DECLARATION

Signature: ________________________

Name: Muhammad Shahril Nizam bin Abdullah

Student ID: 1171102891

Date:

ACKNOWLEDGEMENT

My utmost gratitude and love to my Ma and Abah, also to all my friends who have supported me in this five-years journey in MMU. Thank you for all the support given in getting a scroll of degree from MMU. Finally, I made it until this far. I would never expect that things could turns out like this.

I also would like to give my utmost appreciation towards Ir. Prof. Dr. Hezerul Abdul Karim for accepting as your student to complete this research. I would have not known how to complete all this without your guidance. Thank you as well to Dr. Nouar Aldahoul for the mentorship given towards me in finishing this project. I really appreciate the assistance given. Also, special thanks to all my lecturers. This five years journey in MMU is a fun journey that I would have never forget in my life. Thank you for guiding me to become the greatest person from a kampung guy who knows nothing about all this.

Alhamdulillah, Thank you very much!

ABSTRACT

In the literature, techniques for machine learning and computer vision have been used to analyse violent video content for censorship or monitoring reasons. The prevention of young children and teens being exposed to violent activities that are damaging to viewers' behavioural and mental health is made possible through violence detection.

To categorise films into two categories, violence and non-violence, automatic recognition of violent scenario is important. The deep learning models employed in this work were trained, assessed, and compared using the public dataset, which comprises of two distinct datasets like RWF-2000 and RLVS-2000. One difficulty was found when using this dataset because of the poor video quality, dense crowd, visibility of objects, and transience of action.

In addition, the dataset contents show people engaging in a variety of activities including eating and engaging in sports in diverse settings. The research discussed in this project helps to compare different classification models for the purpose of detecting violence. Light pre-trained convolutional neural networks (CNNs) were used on the domain of violence to extract features. Recurrent Neural Network (RNN) was also included to classify the time series made up of the series of attributes extracted from the video frames. From the experiment, it is proven that EfficientNet B0 outperforms the other lightweight pre-trained CNN with the accuracy of 86.38%, recall of 86.28%, precision of 86.50%, f1-score of 86.39%, and false positive rate of 13.53%.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AEI	Average Energy Images
API	Application Programming Interface
ARM	Advanced RISC Machine
ATM	Automated Teller Machines
BiConvLSTM	Bidirectional Convolutional LSTM
BiLSTM	Bidirectional Long Short-Term Memory
BoW	Bag of Words
CBIR	Content-Based Image Retrieval
CCTV	Closed-Circuit Television
CIFAR	The Canadian Institute for Advanced Research
CLSTM	Convolutional Long Short-Term Memory
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DDR	Double Data Rate
FN	False Negatives
FP	False Positives
FYP	Final Year Project
GPIO	General Purpose Input/Output
GPU	Graphical Processing Unit
GSC	Golden Screen Cinemas
HD	High Definition
HDMI	High-Definition Multimedia Interface
HOF	Histogram Optical Flow
HOG	Histogram of Oriented Gradient
IDE	Integrated Development Environment
IoT	Internet of Things
KLT	Kanade-Lucas-Tomasi
KNN	K-Nearest Neighbours
LAN	Local Area Network
LED	Light Emitting Diode
LMF	Local Motion Feature
LPDDR	Low-Power Double Data Rate
LSTM	Long Short-Term Memory
MAC	Memory Access Cost
MFLOP	Million Floating Point Operations per Second
NLP	Natural Language Processing
OViF	Oriented Violent Flow
PC	Personal Computer
PoE	Power over Ethernet
PWM	Pulse Width Modulation

RAM	Random-Access Memory
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
RIMOC	Rotation-Invariant Feature Modelling Motion Coherence
RLVS	Real-Life Violence Situations
RM	Ringgit Malaysia
RNN	Recurrent Neural Network
SDG	Sustainable Development Goals
SGT	Situation Graph Trees
SPI	Serial Peripheral Interface
SSD	Solid State Drive
SVM	Support Vector Machines
TGV	Tanjong Golden Village
TN	True Negatives
TP	True Positives
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
USD	United States Dollar
VGG	Visual Geometry Group
VID	Violent Interaction Dataset
ViF	Violent Flow

CHAPTER 1: INTRODUCTION

1.1 Overview

We all have a variety of interests that we like engaging in on a daily basis, including watching movies and television shows, playing video games, painting, attending sporting events with friends, and many more. Films provide several advantages in terms of exposure to how one may experience an actor's journey through the played texts at theatres like the GSC and TGV.

Additionally, it is now simple to access via a mobile phone through video streaming services like YouTube, Netflix, Disney+, Viu, and others. Additionally, since the virus spread around the world in 2019, many people have been unable to leave their homes. As a result, viewing movies with friends and family should have been a consideration for individuals of all ages looking for ways to spend their free time.

Even while people of all ages enjoy watching movies, there are some types of content that are inappropriate for viewing, particularly by children and teens. This covers material with strong religious or political overtones, sexual content, or graphic violence, as well as a combination of all three types of material that is subject to censorship [1].

Hence, this project proposes to introduce a method of censoring one of the films censorship types which is Violent Contents. Several comparisons have been made during the experiment stage to evaluate the performance of violence detection in videos using the combination of different pre-trained Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). The combination of these two (2) architectures are crucial since this research will emphasize on video-level classification which involves the extraction of features from video files using pre-trained CNN and RNN.

1.2 Problem Statements

Filmmakers nowadays tends to produce a lot of creative contents for their films, and this is crucial for them to ensure that a lot of people watching this because of the “hype” of the content itself so that it can went viral and massive profit can be made from it. Speaking of which, the creative contents also include the extreme violent scene such as fighting, killing, shooting, and so on and so forth which may affect people’s perspective in their daily life especially towards children and teenagers.

There are various ways that violent contents in film can affect children and teenagers. One of it is the development of children in terms of their behaviour. It is a fundamental for parent to groom and guide their child to become a better person day by day but simple things such as by playing video games as well as watching films could also affect their behavioural development. For example, a new series on Netflix called “All of Us are Dead” contains quite several violent scenes of zombie apocalypse occurred most of the time in the movie which involves zombies biting normal people’s hand, stabbing of zombies using sharp objects, suicide, gores and so on and so forth. As a result, this would lead the children to normalize biting hand of others and trying to play with sharp objects which presumed as a “cool” thing to do based on the series. Hence, censorship is crucial in mitigating the post-watching effect from the series or films.

Besides that, we do understand that adults are currently busy with their daily work even as a housewife where you must take care about house chores and so on and so forth. This makes them unable to monitor their kids at home [2] in terms of what they are watching, hence making their child exposed to unnecessary scene in films while they are watching at the living room, bedroom or even on the phone in the kitchen with their mom while their mom is preparing some food. Hence, this research has been designed on helping to solve these two (2) problems as mentioned and described.

1.3 Project Aim and Objectives

This research aims to provide a system that is reliable for detecting violence situation in movies, as well as for the implementation for detection in public areas. Besides, the project also aims to deploy the said system onto low-cost IoT node, with the help of lightweight pre-trained CNN. Experiments were made to accomplish all the objectives entailed for this research. All objectives were accomplished and can be found as follows:

- To study deep learning image processing technique to assist in detecting violence content inside videos.
- To evaluate various pre-trained Convolutional Neural Network (CNN) models for extraction of features as well as its performance with RNN classifier.
- To perform real-time inferencing for detecting violence in videos using IoT device.

One of the objectives mentioned regarding the IoT device. Raspberry Pi 4 Model B were chosen to be used for real-time violence detection implementation, and it has been purchased together with other several peripherals that are compatible with the IoT device.

1.4 Project Scope

This research started in the final quarter of year 2021 for the duration of approximately one (1) year. Review of literatures were conducted to understand the previous works made from past authors and researchers. Results were also acquired from the past works in detecting the violent content in videos using different approaches such as different utilization of architectures, different datasets acquisition and deployment. Dataset were also obtained in terms of videos collected from various sources. Additionally, all dataset has been collected and categorized according to the different classes available which are violence and non-violence.

Permission has been obtained from the owner of the dataset for the usage in this research.

This project focuses on performing image processing techniques, specifically the feature extraction techniques to extract the necessary features available from the videos, as well as executing the training, validation and testing process using the extracted features on the RNN classifier to perform violence detection in videos. Results were obtained using several data presentation methods, and the most outperformed architecture were chosen for deployment in real-time on the IoT device for inferencing.

Some limitations were found during the research process. Firstly, the lightweight pre-trained CNN contained many layers which constituted to a higher depth value as well as higher number of parameters. Hence, some of the processes involved in this research requires a device with high specifications, which are not available to be obtained by the student. The execution of the project using Python cloud environment such as Google Colab, Kaggle or Paperspace were also unable to be accomplished due to limited amount of memory offered. Hence, the research is altered to run on PC with limited specification, which takes couple of days to completely done on different proposed combinations of model architecture. Besides that, the acquisition of IoT Hardware Device was also a challenging obstacle found by the student whereby there are only limited amount of hardware were sold online by all suppliers. Moreover, the price of the device itself is not cheap whereby it requires several hundred bucks to be used to purchase the said item. The entire project runs within one year with a proper planning which can be found in Table 1 which shows the overall Gantt Chart for this research.

Table 1.1: Gantt Chart for Violence Detection Research

		2021		2022											
No	**Task**	**11**	**12**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	Topic Acquisition	🟦													
2	Journal Reviews		🟦	🟦											
3	Acquisition of Dataset		🟦	🟦											
4	Confirmation of Research Methodology		🟦												
5	Preparation of FYP Research Proposal		🟦	🟦											
6	Submission of FYP Research Proposal				🟦										
7	FYP Research Proposal Presentation						🟦								
8	Experiments					🟦	🟦	🟦	🟦	🟦	🟦	🟦			
9	Collection of Results										🟦	🟦	🟦		
10	Analysis of Results										🟦	🟦	🟦		
11	Preparation of FYP Report												🟦	🟦	🟦

1.5 Report Outline

This report consists of five (5) different chapters, the first one will be discussion regarding the introduction of the research which includes the explanation of the problem statement as well as necessary information in the project scope. Chapter 2 will emphasize on the literature findings that have been made which will contains some information regarding what CNN and the implementation of it as feature extractors for video-level classification is. This chapter will also discuss on the previously implemented techniques in detecting violence in videos which will include the results. Chapter 3 will go deeper into the details of proposed solution for this research which includes the explanation of the dataset acquired, methodology that implemented during the experiment, explanation on different feature extractors that will be used from the available pre-trained CNN, RNN for Violence Detection, as well as the inferencing using video files and real-time using IoT Hardware Device. Chapter 4 will be focusing on the representation of results in terms of Confusion Matrix and Classification Report. Lastly, Chapter 5 will be summarizing the entire research report and this chapter will also report the actual expenses incurred as well as the area of improvements for future research.

CHAPTER 2: LITERATURE REVIEW

This chapter will be focusing on the introduction of Convolutional Neural Network (CNN), the use of pre-trained CNN as the feature extractor, classification in video level, as well as the review of the current implementation techniques of violence detection in videos.

2.1 Introduction of the Convolutional Neural Network

CNN is a subset of deep learning in which it takes an image as the input for performing its processing [3]. An image would have been a great example to represent a binary visual data. The image itself contains a lot of arranged pixels that it is being made up of, and every single value inside each pixel would indicate the brightness of that pixel and it also reflects to what colour that image would have been, based on that value itself. The human brain, in which stores a huge amount of data would perform its processing based on the existing data it has to figure out the image that it is currently looking at. The basic structure that contained in an image such as any curves or visible lines can be easily detected through layers, and it is followed by other inclusive structures such as features in face or the details available in an object. This concept made a computer able to vision how an image would look like, and hence Convolutional Neural Network would be one of the ways to employs this to it.

An architecture of Convolutional Neural Network normally consists of three (3) different layers, namely the convolutional, pooling and fully connected layer. The CNN's fundamental building piece is the convolution layer. Most of the network's computational burden is carried by it.

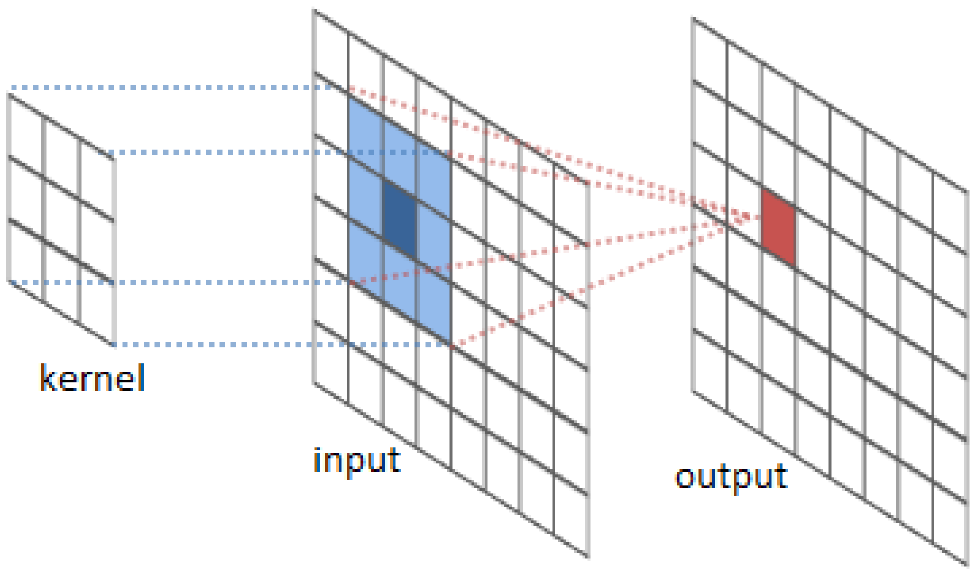


Figure 2.1: Convolution Layer in the CNN Architecture

Convolution layer performs a convolutional operation between two matrixes, where one matrix represents the kernel, a collection of parameters that may be learned and the other matrix represents the constrained area of the receptive field. Although the kernel is deeper than an image, it is spatially smaller as shown in the Figure 2.1. This indicates width and height of a kernel, if a picture has three (RGB) channels, will be spatially modest, yet the depth of it will went up until three channels. The kernel moves across the picture's width and height during the forward pass, creating an image representation of that receptive region. In the end, 2D representation of it known as the map of activation is produced in which it displays the output of the kernel in every location available in the image itself.

By calculating an aggregate statistic from the surrounding outputs, the pooling layer substitutes for the network's output at certain points. This aids in shrinking the representation's spatial size, which lowers the amount of computation and weights needed. Each slice of the representation is subjected to the pooling procedure separately.

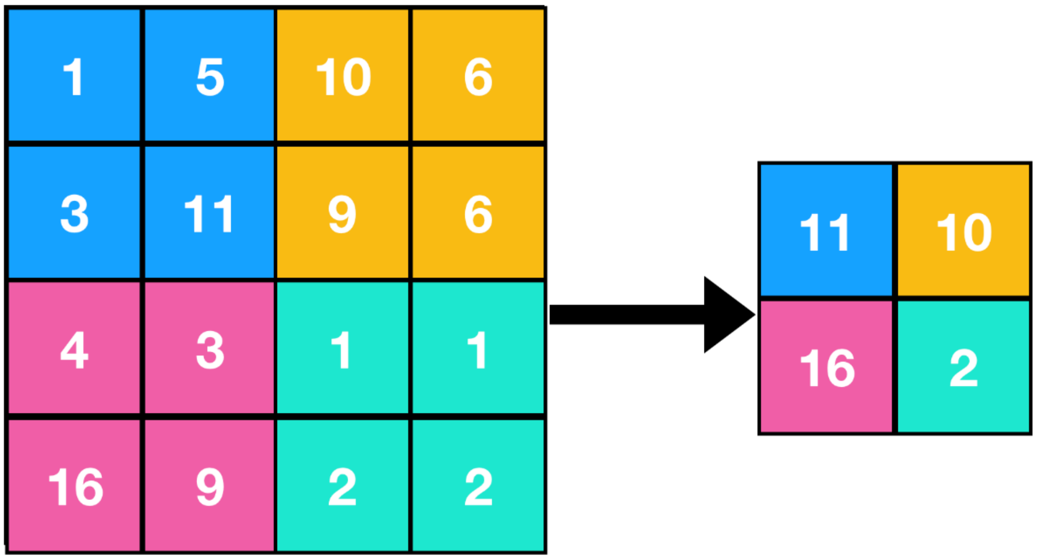


Figure 2.2: The Operation of Pooling in Pooling Layer inside CNN Architecture

A weighted average based on the distance from the centre pixel is one of the pooling functions, along with the average of the rectangular neighbourhood and the L2 norm of the rectangle neighbourhood. However, max pooling, which returns the highest output from the neighbourhood, is the most widely used technique.

$$Output\ Volume\ Size, W_{out} = \frac{Width - Spatial\ Size}{Stride} + 1 \tag{2.1}$$

Fully connected layers, often referred to as linear layers, are frequently employed in neural networks and connect each input neuron to each output neuron.

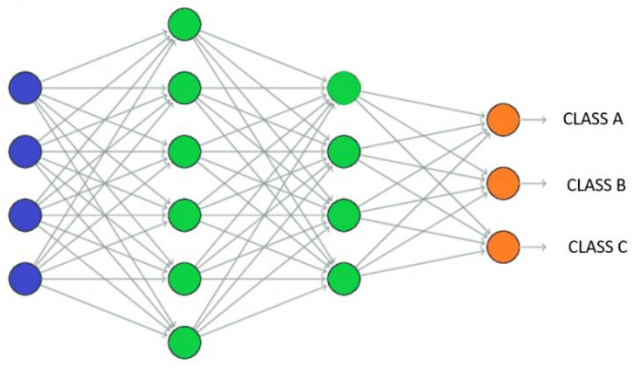


Figure 2.3: Fully Connected Layer in an CNN Architecture

Batch size, number of inputs, and number of outputs are the three factors that characterise a fully connected layer. Forward propagation, computation of the activation gradient, and computation of the weight gradient are all readily stated as matrix multiplications.

2.1.1 Pre-Trained Convolutional Neural Network (CNN) as Feature Extractors

Machine learning models that have already been trained, created, and made accessible by other developers are known as pre-trained models [4]. They are always trained on very big datasets and are typically used to address deep learning-based challenges. They are made accessible by programmers who wish to assist the machine learning community in resolving a comparable issue. Similar to how many developers give back to the community by developing frameworks and packages, some developers also give back by providing a machine learning model, commonly referred to as a pre-trained model. These models are typically created to address both common and extremely complicated challenges. Several pre-trained models will be used in this research such as EfficientNet B0, MobileNetV2, DenseNet-121, and ShuffleNet.

Feature extraction process happens when a set of raw or original data were turned into the number-based features using several processing techniques, without changing the structure or information incurred in the original data representation [5]. This would have made some processes, such as image processing, easier since it would have been hard for a computer or even a machine learning algorithm to understand an image directly, and hence, feature extraction process helps a lot as the mediator between the understanding of human brain and computer processing units. When we want to use pre-trained CNN as the feature extractors, an automatic feature extraction process is used. This research encompasses an automatic procedure of feature extractor in which it does not require any involvement of human during the process. Instead, it used a machine learning algorithm to perform the extraction process which can be easily called during the coding process.

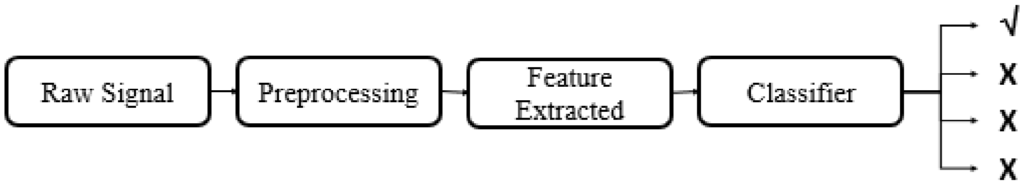


Figure 2.4: Process Schematic in Applying Feature Extraction to Time Series Data for ML Classifier

A machine learning or deep learning algorithm may more readily be consumed when the main distinguishing qualities of signals are identified through feature extraction. Because of the large data velocity and information redundancy, simply training machine learning or deep learning on raw signals frequently produces subpar results.

2.2 Review of Existing Implementation Techniques in Violence Detection in Videos

Violence is defined as any destructive human behaviour in which a person causes injury to himself or to anybody nearby. Violence detection is the process of identifying such behaviours in films or videos obtained via surveillance, and it has recently attracted a lot of attention in the scientific community [6]. For the purpose of detecting aggression, a number of methods have been studied, ranging from machine learning to deep learning models [7]. The research on violence detection in videos using deep learning is highlighted in this portion of the publication. Multiple categorization algorithms have been used in the implementation of violence detection (e.g., traditional machine learning classifiers including KNN, etc.). Various violence detection techniques have been built using traditional machine learning techniques. To detect fights based on the acceleration of spatio-temporal data collected from movies, for instance, the identification of quick fighting has been implemented using motion blob and ellipse detection [8]. A sizable dataset comprising over 1200 clips from movies, hockey, and the ultimate battle challenge was used for the experiment. Although their suggested solution exceeded the state-of-the-art in terms of speed, it performed less accurately than well-known algorithms like KNN and SVM.

It has also been created to identify an object's speed and direction in both crowded and uncrowded environments to detect violent actions using Rotation-Invariant Feature Modeling MOtion Coherence (RIMOC) based on Histogram of Optical Flow (HOF) [9]. Additionally, the identification of the suspected violent individual and the assistance of the security control depend on the recognition of faces during violent incidents. As a result, a research has created a quick facial detection system for use during violent activities [10]. The suggested technique first employs Kanade-Lucas-Tomasi (KLT) face detector for face identification together with Violent Flow descriptor with Horn-Schunck to identify violent activity in videos. Without video frame decoding, statistical theory-based methods have been employed to detect violence. For instance, Region motion descriptor has been employed as a feature extractor in recordings of violence in crowded areas, where the macro block approach has been used to classify the violence [11]. Another statistical technique for violence identification is two-level aggressiveness, which focuses on fight detection to extract motion characteristics and motion's area from video frames. Furthermore, the Bag of Words (BoW) creates the visual characteristics that are employed with the Histogram in order to have an automatic detection of combat scenario [12].

Researchers have looked into the detection of odd activity at the Automated Teller Machines (ATM) in the area of bank security. For instance, the approach of gradient descent using logistic regression has been applied to analyse the posture of users at ATMs. To identify anomalous behaviours based on ATM users' posture, the study leverages data captured by 3D cameras in ATMs. Using joint angles, the study captured postural characteristics. The suggested strategy focuses on the ATM locations, which are often less populated, for the identification of violent acts [13]. Another method of detecting violence employs a three-stage process, starting with the detection of moving objects, followed by the tracking of moving objects and, last, the recognition of activity and behaviour using rule-based categorization and the Gaussian Mixture model [14]. In movies employing object motion detectors, lagrangian fields of direction have been combined with a bag of words to identify violence. Comparing the performance increase to Histogram of Oriented Gradient (HOG) with BoW, the usage of Lagrangian theory has additional value [15]. By analysing moving body parts and acceleration, another method has been utilised to translate animal conflicts onto human combat [16]. The study suggests using the local motion feature (LMF) for segment correlation, motion statistics, and motion analysis. Support Vector Machine (SVM) was used in the experiment to analyse movie and YouTube clips.

SVM is a classifier that is employed in a variety of applications, including violence detection. For instance, Violent Flow (ViF) descriptor has been utilised as an object identification approach in conjunction with SVM as a classifier for violent activities in crowded environments. Using the bag of features approach to extract video characteristics, the study achieved an accuracy of 88% [17]. Additionally, SVM has been utilised as a feature extractor for Convolutional Neural Network (CNN) applications, especially Google Net [18]. Another solution uses an SVM classifier to categorise the violent video characteristics that were extracted using a spatiotemporal grid approach. It also detects objects using spatial pyramids and grids to identify violent acts that are extended from sliding windows and improves the fisher vector [19]. In addition, SVM categorised violent scenarios based on the characteristics of the ViF and Oriented Violent Flows (OViF) descriptors [20]. To categorise HOG and spatiotemporal properties gathered from anomalous periods of visual motion, SVM and the Average Energy Images (AEI) technique for object detection by background removal were utilised [21]. In order to construct Content-Based Image Retrieval (CBIR), a recent study focused on the identification of violent behaviours by pupils using a photo of each student that exhibits a distinct state [22]. The motion areas of the motion regions that the temporal-differencing algorithm had found were located using the Gaussian function. After differentiating people from non-humans using the OMEGA equation's filter, human behaviours were then divided into normal and aberrant violent acts using SVM.

SVM is used to separate abnormalities in movies from typical behaviour since it works best for binary classification. For the objective of feeding video and image data to SVM for violence detection, a variety of feature extraction and object identification algorithms were employed. SVM classifier was used with approaches for feature extraction and object detection. For instance, motion vector method using

Situation Graph Trees (SGT) [25], movement detection with the BoW approach [24], and segments and subsegments representation with HOG & HOF [23]. Object identification using the VGG-f model with ImageNet [26]. Deep learning network designs (i.e., CNN) have been used to study the topic of violence detection, and networks have been used to categorise violent behaviours based on characteristics extracted using convolutional layers [27]. For instance, to identify incidents of aggression and fighting, 3D CNN was utilised to extract features from video frames in the Hockey dataset [28]. Compared to handmade features procedures, the performance accuracy of the model was increased during supervised learning training. Additionally, using data from two open-source datasets, Tokyo 24/7 and Pittsburgh, CNN was used to identify weekly monitored locations where violence occurred [29]. CNN was also utilised to identify violent films based on their audio characteristics. The deep audio features from the 40 Mel Filter Bank were extracted using CNNs. The collected feature blocks of the video dataset were then classified using SVM [30]. Additionally, a CNN and RNN combination was utilised to detect violence from the Violent-Flows, Movies, and Hockey datasets, three public datasets [31]. Convolutional Long Short-Term Memory (CLSTM), which outperforms streaming approaches like ViF and OViF and LSTM, was suggested to be used in the study.

Another CNN-based methodology was used to identify crowd violence and hockey fights by fusing CNN with a trajectory approach, which makes use of both hand-crafted and deep-learned characteristics [32]. Additionally, hybrid features (i.e., learnt and hand-crafted) were applied with both 3D and 2D CNN [33]. Using the Hough Forest classifier on the behave and performed, movie, and hockey datasets, the suggested method categorises the feature retrieved from video frames. To track aggression and altercations during football games, a real-time violence detector was created [34]. The study segmented the films into frames, and the characteristics of the frames were extracted using HOG features. Bidirectional Long Short-Term Memory (BiLSTM) network violent scene recognition in videos was trained using the annotated frames data. Based on information from the past and future, BiLSTM produces an output result [35]. Employing the Violent Interaction Dataset, the developed model was trained (VID). Based on Bidirectional Convolutional LSTM (BiConvLSTM) architecture, Spatiotemporal Encoder was also employed to identify violent activities. The study included datasets from movies, hockey matches, and video game displays [36]. The majority of earlier efforts used modestly sized public datasets, such as Movies with 200 films [37], Hockey Fights with 1000 videos [38], and VFD with 246 videos [39]. On the other hand, datasets like Real-Life Violence Situations (RLVS) with 2000 films [41] and RWF with 2000 videos [40] have not been extensively discussed in literature. Using the RWF dataset, the flow gated network combined 3D-CNNs with optical flow to identify violent scenes in films [40]. Similar to this, the RLVS dataset's categorization of violent films using pre-trained VGG-16 and Long Short-Term Memory [41].

Another type of violence detection method was made by Ravina Dable on 2020 [42]. From her paper, she utilized the CNN architecture to analyse the video that were captured by the CCTV to determine whether the content inside the CCTV contains violent scene or not. By using this way, the author justifies that it is an efficient and convenient way in order to mitigate the negligence among the security staff who are in charge of monitoring the CCTV regarding violent scene, which sometimes can be easily undetected. Additionally, the proposed project from the author also helps the said security staffs who are in charge in the situation whereby there are lesser staff present during work especially during festive season. The author also utilized the 3D Convolutional Network (3D ConvNet) as well as Squeeze-and-Excitation Layer Architectures to run the project. Some future work proposed by the author were to add more datasets to be trained inside the model, adding one Long Short-Term Memory (LSTM) to the network as well as to utilize the rolling window method for prediction purpose.

A hardware implementation of detecting violence using real-time IoT device were made before, and it was authored correspondingly by Nouar AlDahoul back then just in 2021 [43]. The author used a low-cost IoT Node which is Raspberry Pi Board in order to run the CNN-LSTM architecture. Raspberry Pi is a microcontroller that has a limited memory and computational capability. Nevertheless, it is being used by the author together with attached camera to the board where it can be assumed as an embedded violent detection system. This can be used and attached easily to any crowded places for monitoring with the minimal amount of implementation cost. The author fed the classifier’s architecture by using 4000 videos used for both model training as well as model evaluation. Future work proposed by the corresponding author is to implement the same IoT Node to use the recent type of deep CNN such as MobileNet and EfficientNet B0.

Moving on, live detection of violent videos by using dynamic images were made by Ademir Rafael Marques Guedes in 2020 [44]. The author is aims to classify the videos and films in terms of their content whether it does contain inappropriate contents or vice versa and will do necessary actions such as blocking the content towards the specific audience. Besides, the author also has a purpose to aid the security personnel’s performance by having this automated violent detection, and at the same time improving their work performance for their respective areas under surveillance. The author makes use of various datasets such as Hockey Dataset, Movies Dataset as well as Crowd Dataset and several CNN Architectures were also being used such as Dynamic Images Method, Handcrafted Feature Descriptors, Bag of Visual Words and Classification, and Optimization and Combination of Features. The author also proposed to evaluate some other CNN-based descriptors as well as identifying violent location within frame region that contains the violent activities for the future work.

Furthermore, Rasoul Banaeeyan have also conducted similar research for image processing but the content that is being used for detection is nudity materials [45]. The author aims to assist service providers such as the film censorship agency to censor and filter out pornographic materials before the content can be published for public audience. Datasets that have been used were images that we obtained via search engines using the term “Nude Body” for around 1470 images as well as “People” for around 2910 images. The total number of images acquired were 4380 images. The model architecture used for filtering out the nudity material was Residual Learning Convolutional Neural Network or in short, we call it as “ResNet”. The author did put some proposed future work which are to include more variety in terms of nudity which are to classify images into partial nudity as well as some nudity aspects that exist within a complicated picture. Also, the author mentioned to utilize the CUDA-enabled implementation to be used for nudity detection in High Definition (HD) video frames.

Finally, Mi Young Lee [46] focused on detecting violent scenes in movies using novel deep learning approach. The purpose of the author is to omit out the violent scene in movies to ensure that the content itself will not be watched by unwanted audience. There were three (3) different types of datasets used which are Class Fight videos, Hockey Fight videos as well as some videos found from YouTube which sums up to 386 videos. The model architecture used in this project is Fine-Tuning MobileNet Model and the author did propose some ideas for future work which are to utilize sequential learning parameters which is the combination of LSTM and CNN. Besides that, the author also proposed to have a surveillance covered inside smart cities as well as to introduce the embedded vision technologies that has a feature to reduce features size.

2.3 Summary of Existing Implementation Techniques

Based on the existing implementation techniques explained in part 2.2, this subtopic summarized the results obtained from all implementations in terms of the accuracy. The utilized were also mentioned inside the Table 2.1.

Table 2.1: Summary of Existing Implementation Techniques

Author	**Dataset Used**	**Implementation Method**	**Accuracy (%)**	**Ref. No.**
Deniz, O., Serrano, I., Bueno, G., and Kim, T.-K	Hockey Dataset, Movies Dataset	SVM	93.4	[8]
Ribeiro, P. C., Audigier, R., and Pham, Q. C.	Fake train, Real train, Train station, Real life contexts	RIMOC based on HOF	82.20	[9]

Arceda. V. E. M., Fabián, K. M. F., Laura, P. C. L., and Cáceres J. C. G.	Boss dataset, SV	Kanade Lucas and Tomasi (KLT)	97.00	[10]
Xie, J., Yan, W., Mu, C., Liu, T., Li, P., and Yan, S.	VVAR10	SVM	96.20	[11]
Fu, E. Y., Va Leong, H., Ngai, G., and Chan, S.	Hockey game, Action movie	Bag of Words	85.40	[12]
Chaudhary, S., Khan, M. A., and Bhatnagar, C.	Self-defined Dataset	Rule-based Categorization and Gaussian Mixture Model	93.33	[14]
Senst, T., Eiselein, V., Kuhn, A., and Sikora, T.	Hockey fight, Violent Crowd, Violent in Movies, London Riots 2011	LaSIFT	93.12	[15]
Fu, E. Y., Huang, M. X., Va Leong, H., and Ngai, G.	Real life human fights, Hockey games, Action movie	SVM	99.00	[16]
Hassner, T., Itcher, Y., and Kliper-Gross, O.	Hockey dataset, ASLAN dataset	Violent Flows	88.00	[17]
Li, X., Huo, Y., Jin, Q., and Xu, J.	MediaEval15 dataset	SVM	55.09	[18]
Bilinski P., and Bremond, F.	Hockey Fights, Movies, Violent-Flows	SVM	87.00	[19]
Gao, Y., Liu, H., Sun, X., Wang, C., and Liu, Y.	Hockey Dataset, Violent-Flows	ViF + Oriented Violent Flows (OViF)	88.00	[20]
Dhiman, C., and Vishwakarma, D., K.	UR fall detection dataset, KARD	SVM + Average Energy Images (AEI)	95.22	[21]

Al-Nawashi, M., Al-Hazaimeh, O. M., and Saraee, M.	Database for Content-Based Image Retrieval (BCIR)	OMEGA equation	97.00	[22]
Peixoto, B. M., Avila, S., Dias, Z., and Rocha, A.	MediaEval 2013	Bag of Words	73.60	[24]
Song, D., Kim, C., and Park, S-K.	BEHAVE dataset	Situation Graph Trees (SGT)	78.00	[25]
Xia, Q., Zhang, P., Wang, J., Tian, M., and Fei, C.	Hockey Fight, Violent Crowd	VGG-f + SVM	93.25	[26]
Ding, C., Fan, S., Zhu, M., Feng, W., and Jia, B.	Hockey dataset	3D CNN	90.00	[28]
Sudhakaran, S., and Lanz, O.	Violent-Flows, Movies, Hockey Datasets	CLSTM	97.11	[31]
Meng, Z., Yuan, J., and Li, Z.	Crowd Violence Dataset, Hockey Fights Dataset	Fused CNN trajectory	98.60	[32]
Fenil, E., Manogaran, G., Vivekananda, G. N., Thanjaivadivel, T., Jeeva, S., and Ahilan, A.	Violence Interaction Dataset (VID)	Bidirectional LSTM	94.50	[34]
Hanson, A., Pnvr, K., Krishnagopal, S., and Davis, L.	Hockey Fights, Movies	BiConvLSTM	96.96	[36]
T. Hassner, Y. Itcher and O. Kliper-Gross	Violent Crowds	Violent-Flows	85.00	[39]
M. M. Soliman, M. H. Kamal, M. A. El-Massih Nashed, Y. M. Mostafa, B. S. Chawky and D. Khattab	RLVS-2000, Hockey Fight Dataset	VGG-16 + LSTM	90.01	[41]
Ravina Dable	RWF-2000	Squeeze and Excitation Layer	81.00	[42]

Nouar AlDahoul et. al	RWF-2000, RLVS-2000	CNN (Sequential) - LSTM	73.35	[43]
Ademir Rafael Marques Guedes, Guillermo Camara Ch ´ avez	Hockey Dataset, Movies Dataset, Crowd Dataset	SVM	99.80	[44]
Khan SU, Haq IU, Rho S, Baik SW, Lee MY	Violence in Movies, Hockey Fight, Violence Scene Detection	CNN	99.50	[46]

CHAPTER 3: DETAILS OF THE PROPOSED SOLUTION

This chapter will describe further into the datasets used to help in violence detection in videos, the methodology, the hardware setup used to perform the methodology. Besides, this chapter will also detail out the procedures that have been gone through in terms of pre-processing the dataset, information regarding the lightweight pre-trained CNN, feature extractions, classifier's training, inferencing using software as well as the hardware.

3.1 Methodology

Experiments have been conducted using a few procedures which started from processing the videos dataset from the combined RLVS-2000 and RWF-2000 datasets until the evaluation of results based on predictions made from the RNN classifier. The implementation of the model in terms of inferencing were also done using hardware via Raspberry Pi 4 Model B. Figure 3.5 shows the block diagram for violence detection in videos.

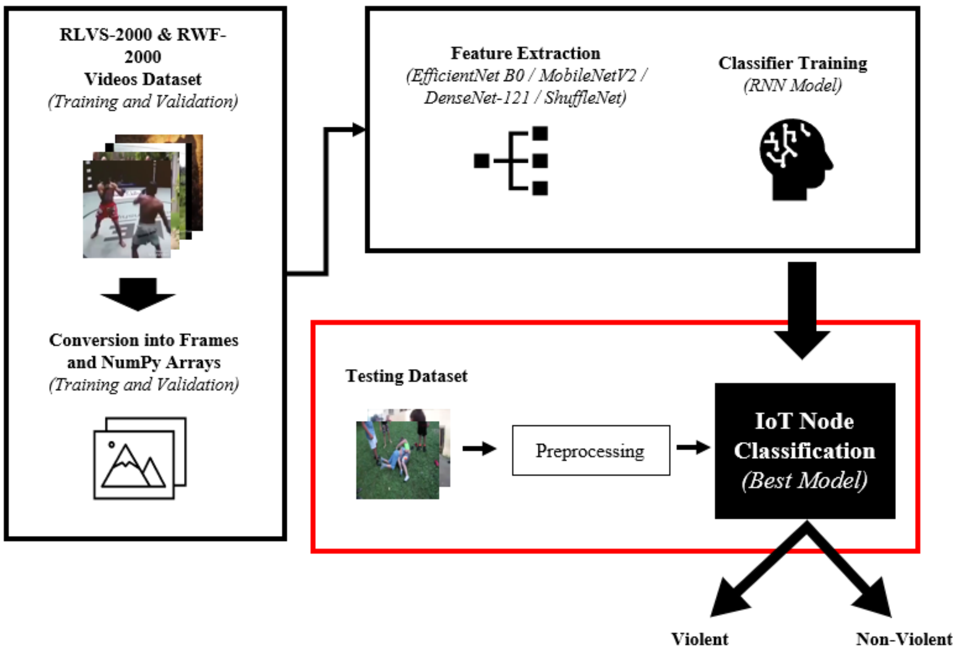


Figure 3.1: Block Diagram for Violence Detection in Videos

3.2 Dataset

There were plenty of datasets available that can be found online. However, for this research, the dataset that have been used were RWF-2000 [40] and RLVS-2000 [41]. Below shows some of the thumbnails available from the RWF-2000 dataset.

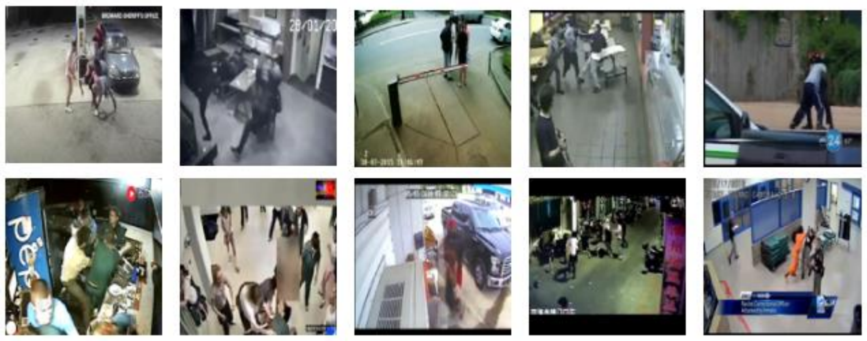


Figure 3.2: A Chunk of RWF-2000 Violence Videos Thumbnails

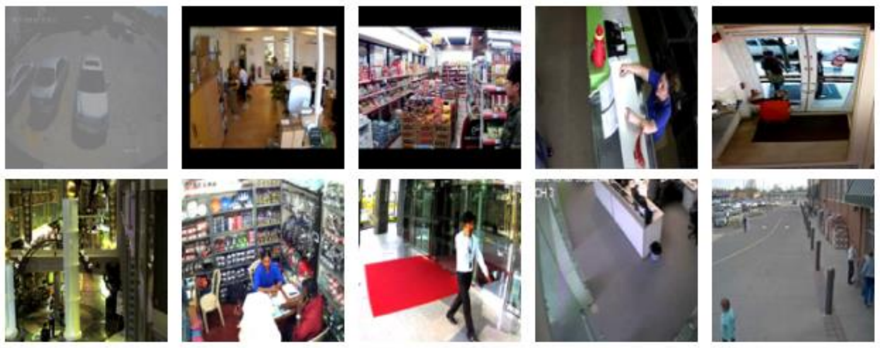


Figure 3.3: A Chunk of RWF-2000 Non-Violence Videos Thumbnails

To carry out this research, two (2) datasets—RWF-2000 and RLVS-2000—were employed. In order to expand the variety of datasets utilised for model training, which improves the accuracy of our model, both of these datasets were pooled into 4000 videos. This dataset includes 2000 movies for each of the two (2) classes—violence and non-violence—which together make up the dataset. To guarantee that more films are fed into the models during the training stage and the model itself can be validated and tested, the videos are then separated into their respective training, validation, and testing datasets with a 3:1:1 ratio.

Both datasets were kept in various ways by their individual authors. Videos from closed-circuit television (CCTV) cameras that represent various real-life events are included in the RWF-2000 dataset. This dataset includes 1000 movies of violence and 1000 videos of non-violence that have previously been categorised into training and validation datasets. The videos that the author personally acquired for the RLVS-2000 dataset, however, include those that were taken from online social media sources. YouTube is a good example. In comparison to RWF-2000 dataset, this dataset contains more diverse video in terms of the nature of the video itself, including some pre-acted combat sequences shot by a group of individuals to show fighting scene. This dataset also includes footage captured by numerous devices, including dash cams, mobile phones, and other devices, in varied contexts. The 1000 violent movies and 1000 non-violent videos from the RLVS-2000 dataset were divided into two (2) distinct groups.

Figure 3.4: A Chunk of RLVS-2000 Violence Videos Thumbnails

Figure 3.5: A Chunk of RLVS-2000 Non-Violence Videos Thumbnails

Ultimately, this research combined two (2) of the aforementioned datasets, resulting in a total of 4000 videos being used, of which 3200 were used for training and validation and 800 were used for testing. This was done in an effort to increase variation as well as to improve the accuracy of detection. Table 3.1 displays an overview of videos in dataset that were divided into training, validation, and testing.

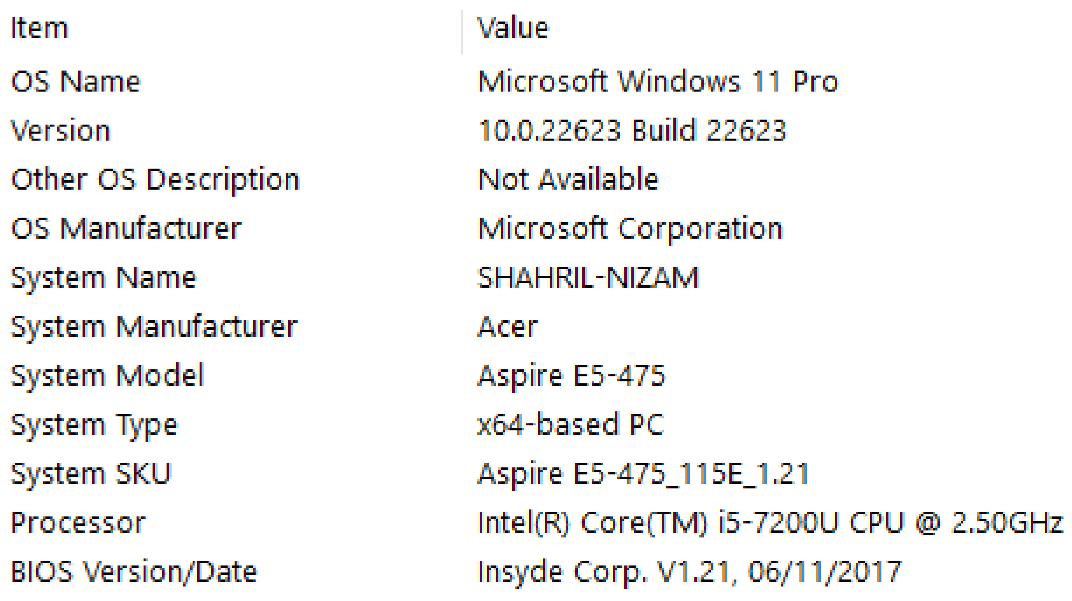
Table 3.1: Overview of the Combined Dataset

Dataset	No. of Videos in Dataset	Category	Total
RWF-2000	2000	Training	1280
		Validation	320
		Testing	400
RLVS-2000	2000	Training	1280
		Validation	320
		Testing	400
Total Number of Videos in the Combined Dataset			4000

3.3 Hardware Setup

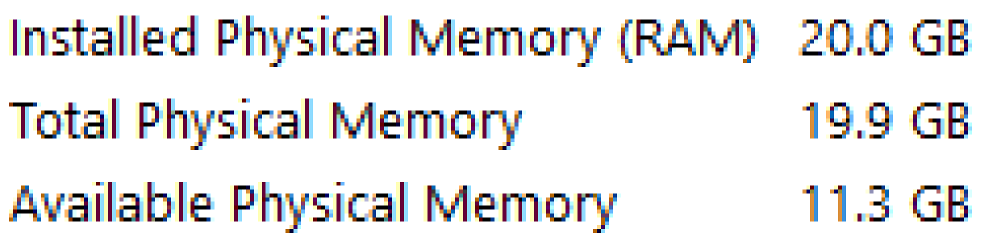
The experiment was conducted in an offline environment with limited specifications. Several free cloud-based environments made available online were not usable due to the limited memory offered, for instance, Google Colab, with only 16 GB free GPU offered in its free features [47]. Hence, since our experiment is utilizing a huge amount of memory allocation, another simpler and zero-cost approach was used in order to mitigate the insufficient memory issue. Additionally, there are no Graphical Processing Unit (GPU) was used during the entire experiment.

A combination of local RAM with Virtual Paging Memory were used to run the dataset pre-processing, model training as well as inferencing on the same device. The device runs on Windows 11 with Intel ® Core ™ i5-7200U CPU @ 2.50GHz. A 20 GB DDR4 RAM was used during the entire process with the pre-allocated 80,000 MB Virtual Memory that were taken from a portion of the WD Green ™ SATA SSD M.2 2280.



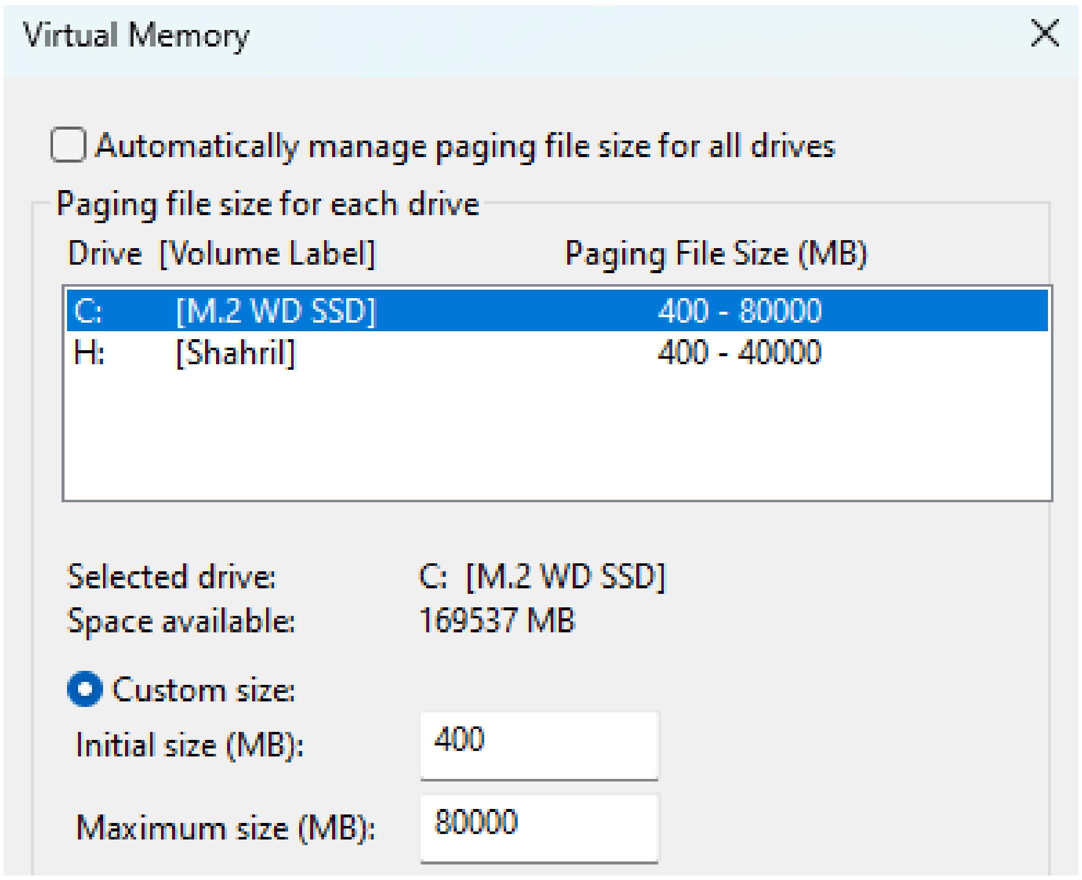
Item	Value
OS Name	Microsoft Windows 11 Pro
Version	10.0.22623 Build 22623
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	SHAHRIL-NIZAM
System Manufacturer	Acer
System Model	Aspire E5-475
System Type	x64-based PC
System SKU	Aspire E5-475_115E_1.21
Processor	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
BIOS Version/Date	Insyde Corp. V1.21, 06/11/2017

Figure 3.6: Processor Information



Installed Physical Memory (RAM)	20.0 GB
Total Physical Memory	19.9 GB
Available Physical Memory	11.3 GB

Figure 3.7: Memory Information


Virtual Memory

[ ] Automatically manage paging file size for all drives

Paging file size for each drive

Drive	[Volume Label]	Paging File Size (MB)
C:	[M.2 WD SSD]	400 - 80000
H:	[Shahril]	400 - 40000

Selected drive: C: [M.2 WD SSD]
Space available: 169537 MB

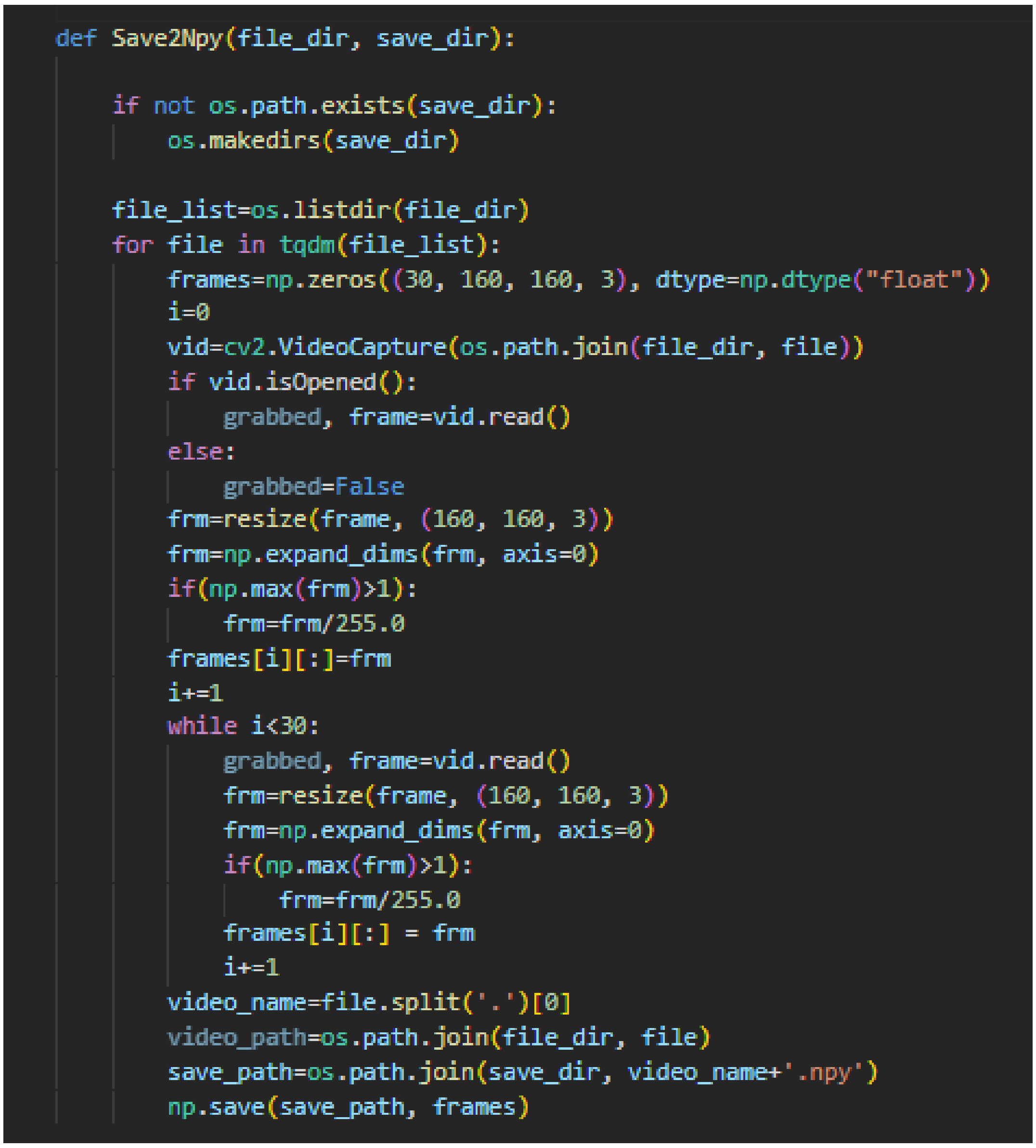
[✓ Custom size:]
Initial size (MB): 400
Maximum size (MB): 80000

Figure 3.8: Virtual Memory Allocation Information

The execution of experiment was done using a code editor named Visual Studio Code [48]. Debugging, task execution, and version control are supported by the simplified code editor Visual Studio Code. It tries to give developers only the tools they require for a short cycle of code-build-debugging and leaves more sophisticated processes to IDEs with more features, like Visual Studio IDE.

3.4 Dataset Pre-processing

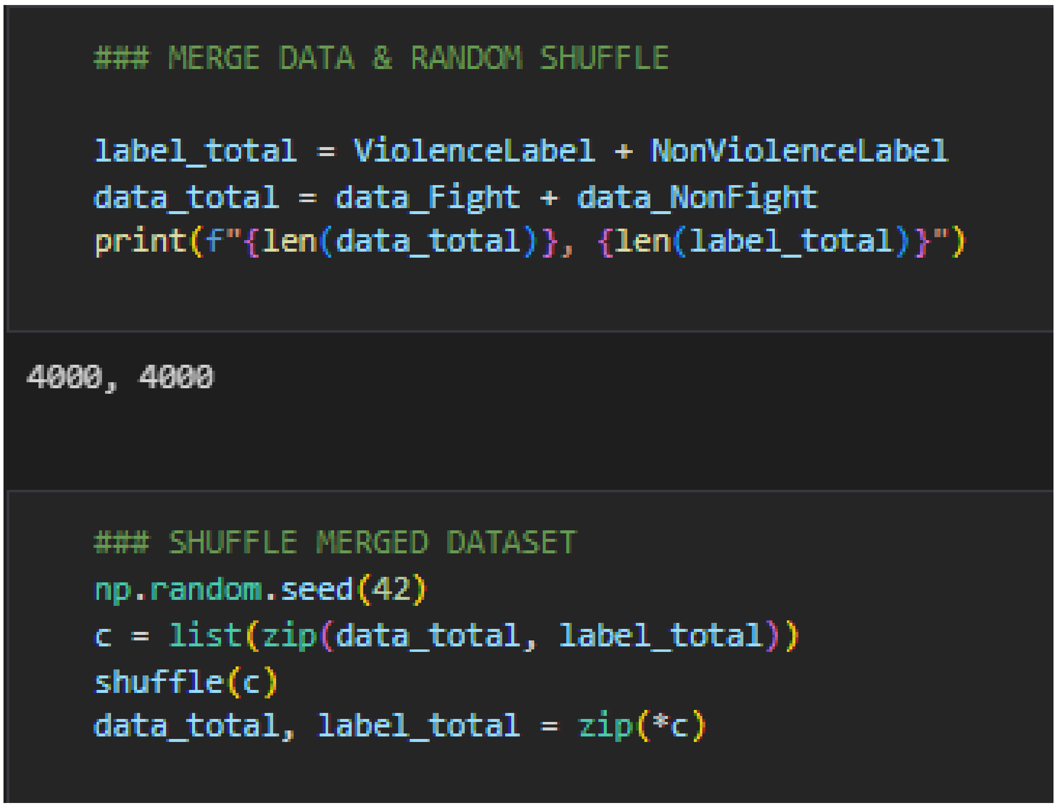
There are three (3) parts involved in the dataset pre-processing which includes the conversion of videos into frames, the conversion of the frames into arrays of NumPy, as well as the dataset splitting into training, validation and testing sets. The first two (2) processes are crucial to ensure the models, both of the feature extractors and the classifier, understands the content of the dataset since the only way for both models to read video contents are through the arrays of NumPy.



```python
def Save2Npy(file_dir, save_dir):

    if not os.path.exists(save_dir):
        os.makedirs(save_dir)

    file_list=os.listdir(file_dir)
    for file in tqdm(file_list):
        frames=np.zeros((30, 160, 160, 3), dtype=np.dtype("float"))
        i=0
        vid=cv2.VideoCapture(os.path.join(file_dir, file))
        if vid.isOpened():
            grabbed, frame=vid.read()
        else:
            grabbed=False
        frm=resize(frame, (160, 160, 3))
        frm=np.expand_dims(frm, axis=0)
        if(np.max(frm)>1):
            frm=frm/255.0
        frames[i][:]=frm
        i+=1
        while i<30:
            grabbed, frame=vid.read()
            frm=resize(frame, (160, 160, 3))
            frm=np.expand_dims(frm, axis=0)
            if(np.max(frm)>1):
                frm=frm/255.0
            frames[i][:] = frm
            i+=1
        video_name=file.split('.')[0]
        video_path=os.path.join(file_dir, file)
        save_path=os.path.join(save_dir, video_name+'.npy')
        np.save(save_path, frames)
```

Figure 3.9: Conversion of Videos to NumPy arrays

4000 videos were stored into a single folder, separated into subfolders according to its related classes such as violence and non-violence. The videos were converted into frames and resized into the size of 160 x 160 pixels. The frames were then converted into arrays of NumPy and each video files were saved as a NumPy file after being converted to frames and resized according to the mentioned size. The respective labels were also created in terms of NumPy arrays, whereby [0,1] are known as violence and [1,0] for non-violence. All series of frames for all videos were then being appended into its single files, two separate files for arrays of videos, two separate files for arrays of labels. The two there represents the number of classes each array are holding which are violence and non-violence.



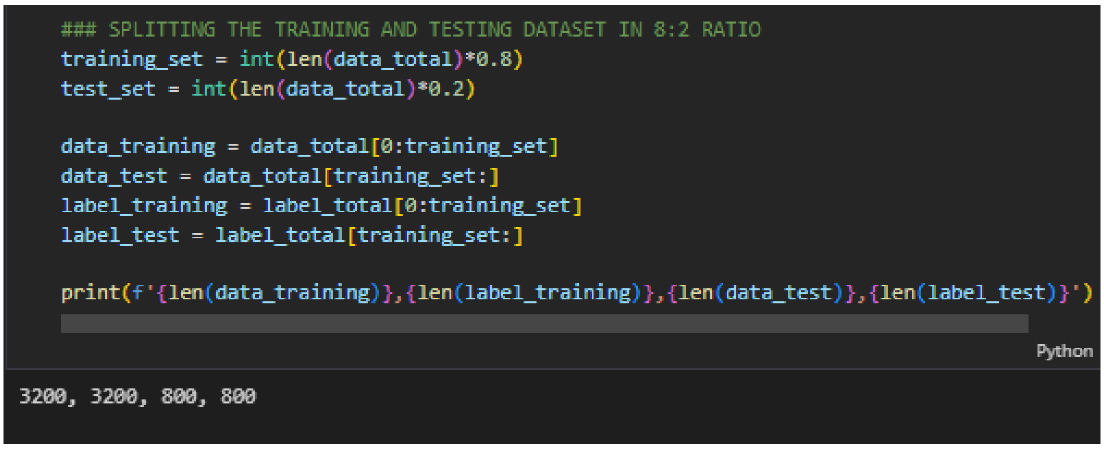
```
### MERGE DATA & RANDOM SHUFFLE

label_total = ViolenceLabel + NonViolenceLabel
data_total = data_Fight + data_NonFight
print(f"{len(data_total)}, {len(label_total)}")
```

```
4000, 4000
```

```
### SHUFFLE MERGED DATASET
np.random.seed(42)
c = list(zip(data_total, label_total))
shuffle(c)
data_total, label_total = zip(*c)
```

Figure 3.10: Dataset Merge and Dataset Randomization

The dataset was merged for both classes which combined the violence and non-violence classes altogether. This is crucial for us to do random shuffling in the dataset, in which helps the classifier to learn the data without looking at any specific sequences. Instead, random data were fed into models which increases the variety of sequences for the classifier to look at, in which in the end, improves the precision of classifier in performing predictions during the implementation stage. The randomization shuffling was done together for both data and labels to ensure that the randomized data matched with the labels, which to be used for verification during the evaluation stage.



```
### SPLITTING THE TRAINING AND TESTING DATASET IN 8:2 RATIO
training_set = int(len(data_total)*0.8)
test_set = int(len(data_total)*0.2)

data_training = data_total[0:training_set]
data_test = data_total[training_set:]
label_training = label_total[0:training_set]
label_test = label_total[training_set:]

print(f'{len(data_training)},{len(label_training)},{len(data_test)},{len(label_test)}')
```

```
3200, 3200, 800, 800
```

Figure 3.11: Splitting Dataset into Training and Testing Sets

Splitting dataset into appropriate ratio is crucial in ensuring the classifier obtained enough data to reflect good accuracy and maintaining minimal loss. Hence, the violence detection dataset was split into training and testing sets, whereby 80% of the overall dataset were taken as the training set which totals up to 3200 videos, and the rest are for testing.

3.5 Introduction to Lightweight Pre-Trained CNN

Several lightweight pre-trained Convolutional Neural Network (CNN) models were chosen due to small numbers of parameters which known to be useful for implementation in a lower specification device such as IoT device. The models chosen were EfficientNet B0, MobileNetV2, DenseNet-121 and ShuffleNet.

3.5.1 EfficientNet B0

A compound coefficient is used by the EfficientNet [49] CNN design and scaling approach to reliably scale the parameters, such as resolution, width, and depth. In contrast to conventional technique, which scales the variable freely, the EfficientNet scaling approach uniformly scales the specified parameters using a set of default and constant scaling coefficients. For instance, to utilise 2^N times more processing power, we may simply increase the network's depth α^N, width β^N, and picture size γ^N, where all the symbols used (α, β, γ) are the constant coefficients that were found using a little grid search on the original small model. This model uses a compound of coefficients to consistently scale the network's resolution, depth, and width. This compound scaling method is justified by the fact that larger input pictures demand additional layers within the architecture to increase the network's receptive area and more channels to capture more minute patterns on the larger picture.

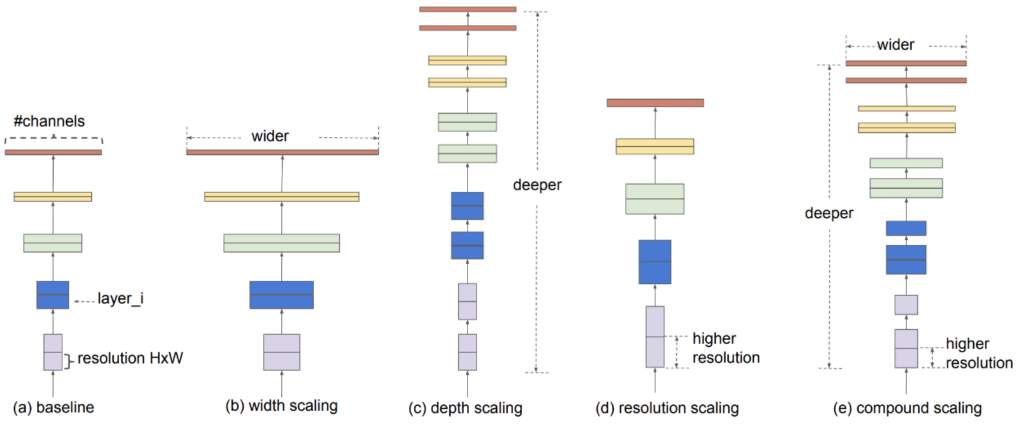


Figure 3.12: The Model Scaling of EfficientNet [49]

The squeeze-and-excite blocks and the inverted bottleneck leftover MobileNetV2 blocks form the basis of the EfficientNet-B0 network. EfficientNets performs well and achieves a good accuracy while utilising orders of orders of magnitude less parameters on various datasets, including as the Flowers (98.8%), CIFAR-100 (91.7%), and the other three additional datasets.

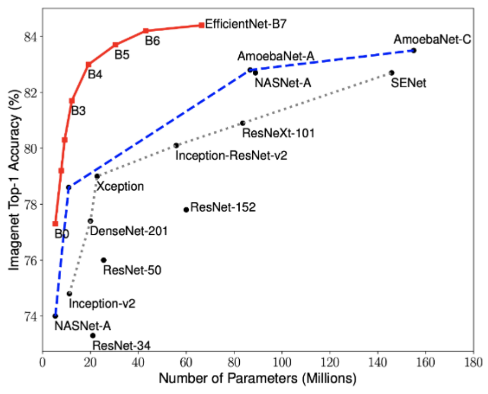


Figure 3.13: Performance of EfficientNet [49] over the other pre-trained Models

3.5.2 MobileNetV2

MobileNetV2 [50] improves the state-of-the-art for mobile visual recognition in terms of classification, object identification, and semantic segmentation. Compared to MobileNetV1, it is a significant improvement. MobileNetV2 was made available with the TensorFlow-Slim Image Classification Library. Based on the principles of MobileNetV1, it was built using depth wise separable convolution as efficient building blocks. Two additional features added by V2 to the system are the linear bottlenecks between the layers and the shortcut connections between the bottlenecks.

The inner layer of the model contains the model's capacity to transition between lower-level ideas like pixels and higher-level descriptors like picture categories, whilst the bottlenecks represent the model's intermediary inputs and outputs. Finally, shortcuts, like typical residual connections, provide faster training and enhanced accuracy.

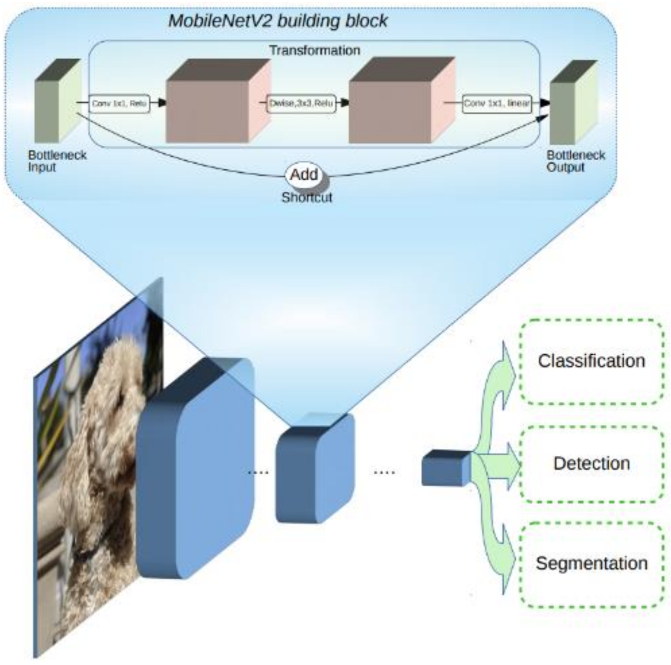


Figure 3.14: Overview of MobileNetV2 [50] Model Architecture

3.5.3 DenseNet-121

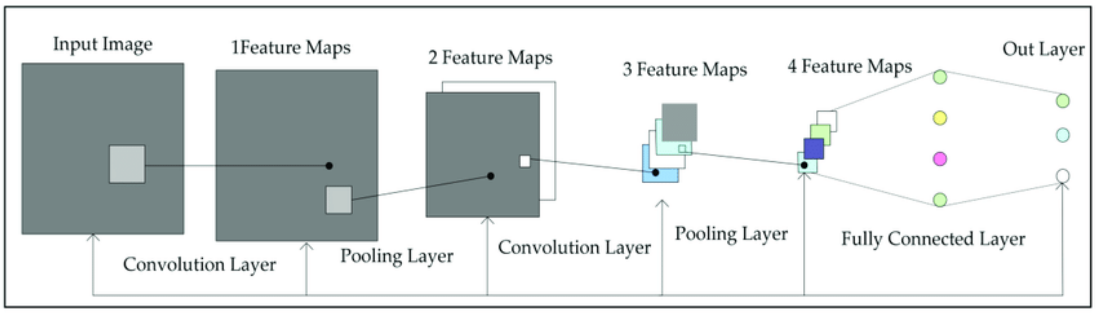


Figure 3.15: The Layers inside the Architecture of DenseNet-121 [51]

The DenseNet-121 model is one of the DenseNet family's image classification models. All of the DenseNet models have previously completed pre-training on the ImageNet picture dataset. The output of the previous convolutional layer is the sole one supplied to the first input layer in DenseNet-121 [51], the traditional feed-forward CNN. This convolutional layer then produced the output feature map and sent it to the convolutional layer that came next. As a result, there are "L" straight connections going from one layer to the next for each layer.

When the CNN had additional layers or had descended farther into the architecture, the "vanishing gradient" was established. This suggests that while information does extend as it moves from the input to the output layers, some information may also "disappear" or "vanish," which reduces the network's ability to successfully undergo the learning process. By streamlining the connections between layers and altering the traditional CNN design, DenseNets alleviate this issue. An architecture in which each layer is directly connected to every other layer is referred to as a "Densely Connected Convolutional Network."

3.5.4 ShuffleNet

A straightforward yet very effective CNN design called ShuffleNet [52] was developed primarily for devices with little memory and processing power, or 10-150 MFLOPs (Mega Floating-point Operations Per Second). Because of how well it worked in trials and how much it gained in popularity, prestigious universities have included it into their curricula. Then, Ningning Ma et al. from Megvii and Tsinghua University produced ShuffleNet V2, which also considers Memory Access Cost (MAC), the degree of parallelism, and platform dependency in addition to MFLOPs (here, MFLOPs are direct measurements). They offer new guidelines for a mobile network design that is faster, more accurate, and flawlessly runs on both GPU- and ARM-based hardware. The most current versions of ShuffleNet, including the ShuffleNetV2 Large and ShuffleNetV2 Extra Large architectures, are deeper networks with greater trainable parameters and FLOPs to accommodate devices with more computing and storage capability.

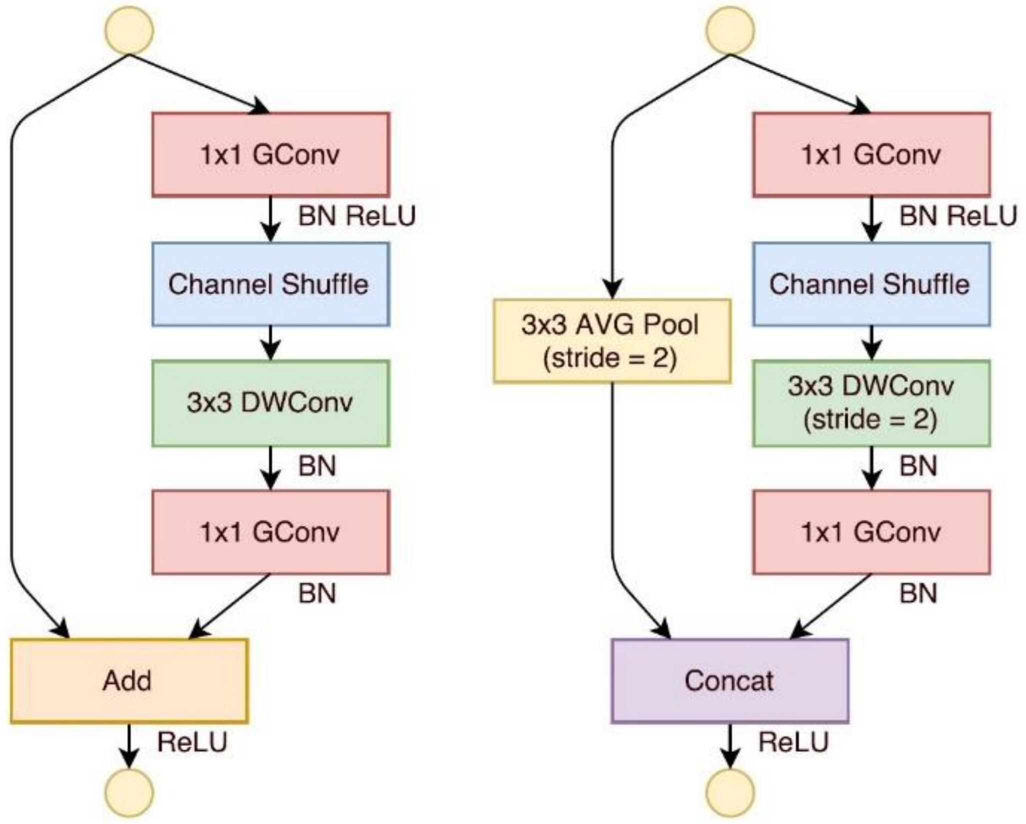


Figure 3.16: The Shuffle Unit inside the Architecture of ShuffleNet [52]

A typical ShuffleNet with shuffled action channel and convolutional pointwise group is seen in the diagram above. In order to accommodate the dimensions of the output feature representations, the network with stride=1 is displayed in the left one and the network with stride=2 and concatenation is utilised in the right one.

Dimension matching is utilised with the skip connection and the second pointwise operation. There are two pointwise operations in it. The second pointwise convolution is not subjected to the shuffle operation since doing so produces results that are comparable.

3.5.5 Feature Extraction using Lightweight Pre-Trained CNN

Before the input files (the arrays of NumPy) were trained, validated, and tested, feature extraction shall take place first in order to extract the necessary features available in the video files. The feature extraction process was executed by passing the input data arrays into the pre-trained CNN models separately, and different output shape from the extracted features were obtained according to the respective pre-trained models. The feature extraction process takes slightly different time from one pre-trained CNN to another according to its depth, or in other words, the higher the number of layers available in a pre-trained CNN, the higher the time taken for the feature extraction process to complete.

Table 3.2 shows the information on different feature extractors in terms of the output size shape it provides after the arrays of data has been passed, the depts, as well as the number of parameters for each feature extractors.

Table 3.2: Information of Feature Extractors (Pre-trained CNN)

Feature Extractor	Depth	No. of Parameters	Input Shape	CNN Output Shape
EfficientNet B0	132	4,049,571	160x160x3	5x5x1280
MobileNetV2	105	2,257,984		5x5x1280
DenseNet-121	242	7,037,504		5x5x1024
ShuffleNet	50	969,258		1x1x576

Based on Table 3.2, EfficientNet B0 model has 132 layers that can be used to extract the features available from the input video file, with more than 4 million parameters. The output shape of 5x5x1280 were obtained from the Conv2D, Batch

Normalization and Activation layers, the last three layers after removing the imagenet weights from the EfficientNet B0 architecture.

It has a same output shape with MobileNetV2. However, MobileNetV2 has a smaller amount of depth compared to EfficientNet B0 architecture. Besides that, the last layer of MobileNetV2 consist of the rectified linear unit or ReLU activation layer which finalized the output layer for extraction of feature for the video files using MobileNet V2 model.

DenseNet-121 has the largest number of parameters as well as the number of depths in its architecture compared to the three other models. DenseNet-121 also used the ReLU activation layer, same as MobileNetV2 architecture. However, it produces a different output shape, in which the shape is 5x5x1024.

ShuffleNet, however, is the smallest model available used in this research. It has only 50 depths or number of layers which constitute towards less than 1 million number of parameters available to be used during the feature extraction. The output shape is also the smallest with the shape of 1x1x576 according to its final layer after the model has been rescaled to accept the desired input shape of 160x160x3.

3.6 Recurrent Neural Network for Violence Detection in Videos

A sort of artificial neural network called a recurrent neural network (RNN) [53] uses sequential data or time series data. For ordinal or temporal issues, such as language translation, natural language processing (NLP), speech recognition, and image captioning, these deep learning algorithms are frequently applied. Recurrent neural networks (RNNs) learn from training data similarly to feedforward and convolutional neural networks (CNNs). Their "memory," which allows them to use data from earlier inputs to affect the present input and output, sets them apart from other systems. Recurrent neural networks' outputs depend on the previous items in the sequence, in contrast to standard deep neural networks' assumption that inputs and outputs are mutually exclusive. Unidirectional recurrent neural networks are unable to anticipate future occurrences, even if they would be useful in predicting how a series would turn out.

Recurrent networks are distinguished by the fact that each layer of the network uses the same parameters. Recurrent neural networks share the same weight parameter inside each layer of the network, in contrast to feedforward networks, which have distinct weights across each node. However, to support reinforcement learning, these weights are still modified using the techniques of backpropagation and gradient descent.

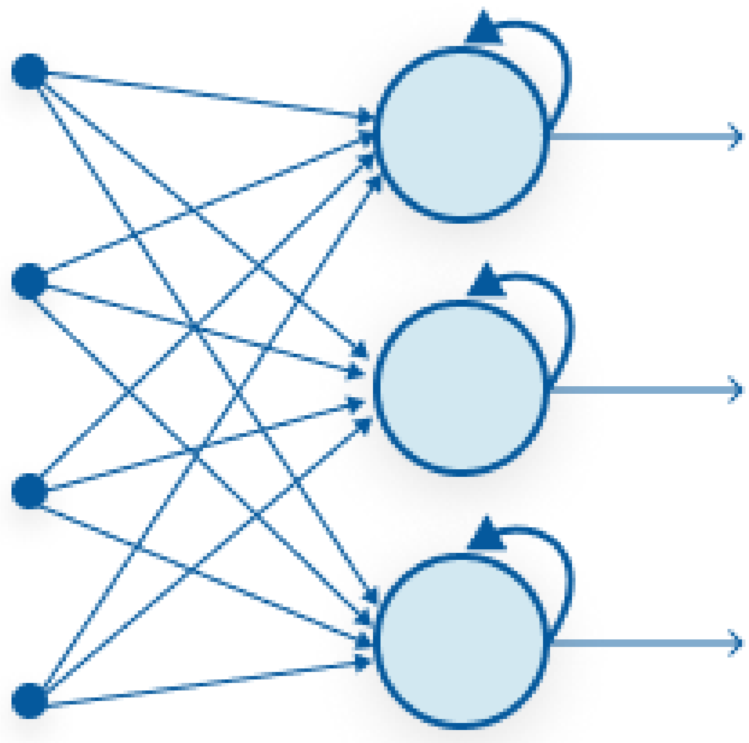


Figure 3.17: A Simple Visualization of an RNN Structure

This research uses a combination of CNN and RNN to perform violence detection in videos. CNN was used as feature extractors, meanwhile RNN was used as the classifier or the model that will determine the prediction of violence occurrence in videos. Experiment was conducted using different feature extractors to determine its suitability while performing predictions with the RNN classifier. Given that different feature extractors were used, a constant RNN model was used with its layers being defined arbitrarily to ensure a minimal depths and number of parameters to took place, as well as high accuracy obtained during the model training, validation, and testing processes.

Hence, there are several parameters as well as the architecture were defined arbitrarily, in order to achieve the goal of this research. Some different layers were used in the architecture, with different number of neurons included in each layer, such as the subset of RNN layers itself which known as the Long Short-Term Memory (LSTM) layer, Dense layer, Rectified Linear Unit (ReLU) activation layer as well as the SoftMax activation layer.

The model was compiled using categorical crossentropy loss with adam optimizer at a default learning rate of 0.01. The “accuracy” metrics were focused for monitoring purpose. Finally, the model accepts the product of the 3-dimensional input from the extracted features’ output shape respectively, and the model were set to have 50 number of epochs with 30 batch size. Table 3.3 and Table 3.4 summarizes the explanation on the RNN classifier model in terms of architecture and hyperparameters.

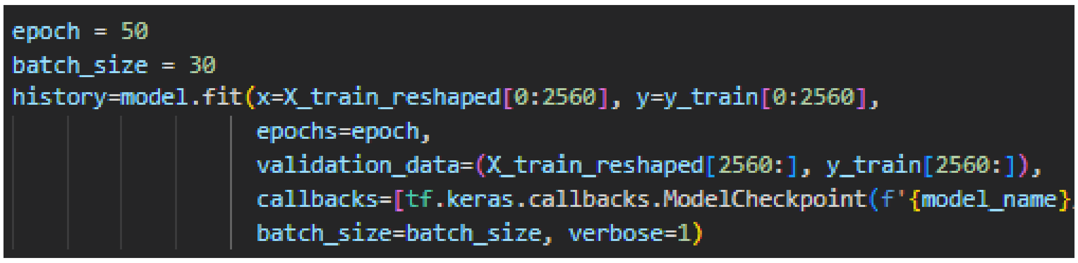
Table 3.3: RNN Classifier Architecture

Layer	Type of Layer	Output Shape	No. of Parameters
LSTM	LSTM	(None, 10)	1280440
Dense	Dense	(None, 512)	5632
ReLU Activation	Activation	(None, 512)	0
Dense	Dense	(None, 1024)	525312
ReLU Activation	Activation	(None, 1024)	0
Dense	Dense	(None, 2)	2050
SoftMax Activation	Activation	(None, 2)	0

Table 3.4: RNN Classifier Hyperparameters

Monitored Loss	: Categorical Crossentropy
Optimizer	: adam
Monitored Metrics	: accuracy
Learning Rate	: 0.01
Number of Epochs	: 50
Batch Size	: 30
Input Size	: Product of CNN Output Shape

The extracted features were fitted inside the model whereby the first 80% (2560 videos) of the training set were used by the model for training, and the rest were used for validation in each epoch. The model checkpoints were taken to avoid any potential faulty coming from the machine used for training since it only has a decent specification for normal use. Verbosity level was set to 1 so that the real-time progress can be seen throughout every epoch.



```
epoch = 50
batch_size = 30
history=model.fit(x=X_train_reshaped[0:2560], y=y_train[0:2560],
                  epochs=epoch,
                  validation_data=(X_train_reshaped[2560:], y_train[2560:]),
                  callbacks=[tf.keras.callbacks.ModelCheckpoint(f'{model_name}
                  batch_size=batch_size, verbose=1)
```

Figure 3.18: Model Fitted for Training and Validation

The results were obtained in terms of accuracy and loss for both training and testing batches. Besides that, the results were also obtained in terms of confusion matrix and classification report from passing the test batches into the RNN classifier. The results will be further discussed in Chapter 4.

3.7 Real-Time Violence Inferencing using IoT Hardware Device

The classifier was trained with decent accuracy and the best combination of model was selected to perform inferencing. The process of employing a trained neural network model to produce a prediction is known as inferencing. On the other hand, model training describes the development of the aforementioned model or machine learning algorithm using a training dataset, in this case our violence detection model. The method of inferencing that is going to be deployed using the best model will be implemented using IoT hardware device which is Raspberry Pi 4 Model B. It is a microcontroller with a graphical user interface which enables user to perform work, research, or any other matters regarding the device in an easier way.

3.7.1 Introduction to Raspberry Pi 4 Model B

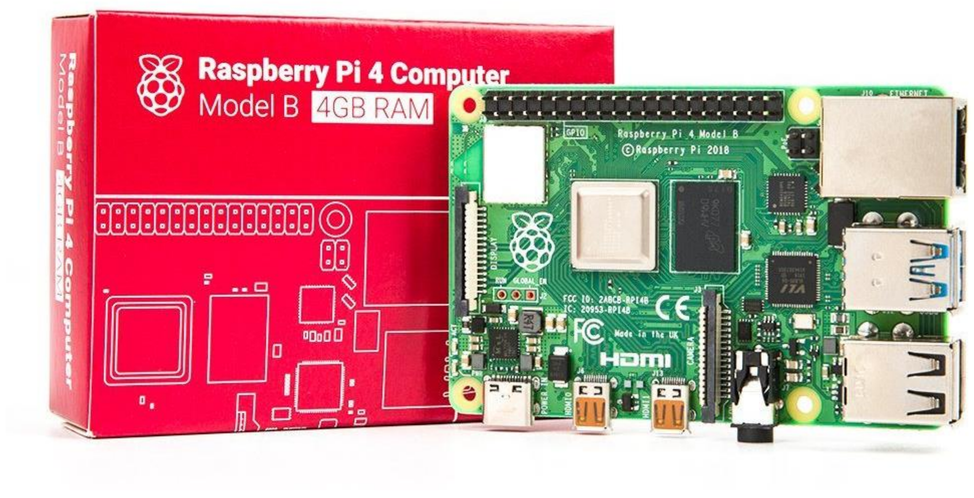


Figure 3.19: An Example of Raspberry Pi 4 Model B

Raspberry Pi [55] is known as a one-board and all-in-one computer that was developed by a non-profit organization based in the United Kingdom known as the Raspberry Pi Foundation. The foundation itself is moving vastly with a mission of empowering literacy of computer and providing an easy access towards the education specifically in the computer science field. The organization itself has developed plenty versions of Raspberry Pi since its debut made in the year 2012.

The first version of Raspberry Pi was featured with only single-core CPU with 700MHz frequency and RAM of 256MB, which is the lowest Pi specs among all the available Pi out there. The most recent developed Pi is far more advanced in which in includes a quad-core CPU at 1.5GHz frequency and the RAM of 8GB. It is known to be a low-cost computer since it only costs the consumer around USD100 (nearly around RM440) to own one. The affordable price offered enabled the industrial practitioners, researchers, students, and enthusiasts to develop their coding skills, creating projects for social improvements, home automation using Internet of Things (IoT), performing edge computing and even for implementation in the industrial level for their businesses.

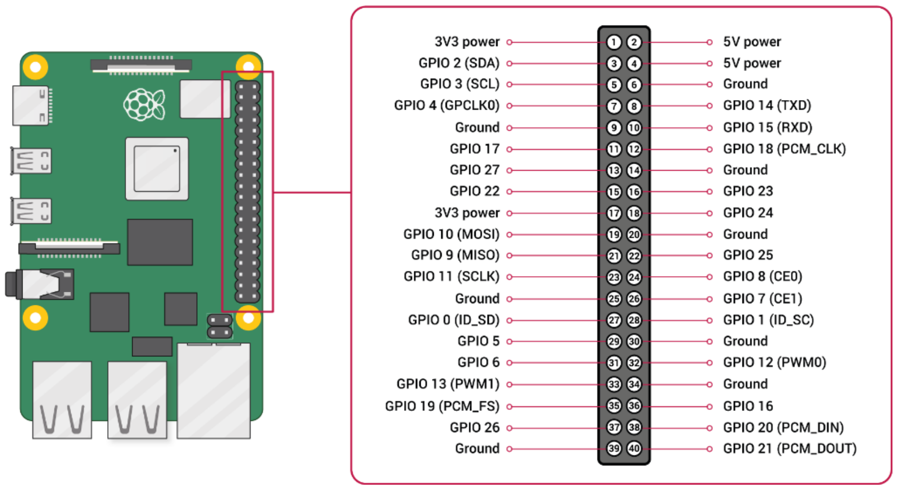


Figure 3.20: The Pinout of Raspberry Pi 4 Model B

There are 40 pins total on the Raspberry Pi 4 GPIO Pinout: 26 GPIO pins, two 5V pins, two 3V3 pins, and seven ground pins (0V). The RPI 4's GPIO pins may provide PWM output, and the board supports the serial communication protocols SPI, I2C, and UART. In June 2019, the Raspberry Pi Foundation unveiled its most recent board, the Raspberry Pi 4 Model B. This model features the most recent quad-Core, 64-bit, 1.5GHz Broadcom 2711 Cortex A72 CPU. This CPU performs 90% better than the previous generation while using 20% less electricity. Three alternative LPDDR4 SDRAM configurations for the Raspberry Pi 4 model are available: 2 GB, 4 GB, and 8 GB. Other new features of the board include hardware video decoding at up to 4Kp60, dual-channel 2.4/5.0GHz wireless LAN, genuine Gigabit Ethernet, two USB 3.0 ports, Bluetooth 5.0, and PoE functionality. Dual display supports up to 4K resolutions is also provided by a pair of micro-HDMI connections (via a separate PoE HAT board).

3.7.2 Peripherals and Circuitry

Since Raspberry Pi was used to perform violence detection in real-time, there were several other peripherals used together with the raspberry pi, which acts as the input to capture all the images as well as some small components that acts as the actuator which performed certain actions after a violence situation was detected.

The peripherals used for violence detection in videos via Raspberry Pi 4 Model B were USB Web Camera as well as RGB Light Emitting Diode (LED). A typical USB Web Camera were used for capturing the real-time videos available from the surrounding. It reduces the cost of implementation by not needing to have a higher quality camera to be used during the inferencing stage since a decent quality of camera that is typically used for video conferencing is already enough to perform violence detection.

The RGB LED was used for the microcontroller to signify whether the output of an instantaneous detection was actually a non-violence (green colour) or violence (red colour) situation. The full circuitry of the violence detection system is shown in Figure 3.21.

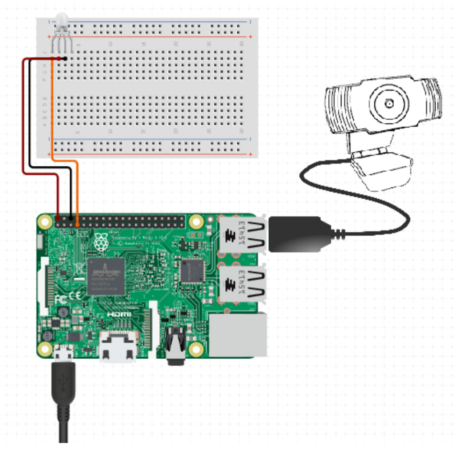


Figure 3.21: Violence Detection System Circuitry

CHAPTER 4: DATA PRESENTATION AND DISCUSSION OF FINDINGS

This chapter will cover on the presentation of data obtained from the experiments, or in other words, the predictions result in terms of accuracy and loss over the test set, discussion of results in terms of confusion matrix, as well as in terms of classification report. Additionally, this section will also emphasize on the real-time implementation and the IoT Dashboard that was built via Thingspeak.

4.1 Presentation and Discussion of Results in terms of Accuracy and Loss

The training and validation phase of the model were done using 50 epochs at the batch size of 30, and the results for each epoch were obtained in terms of training accuracy, validation accuracy, training loss and validation loss.

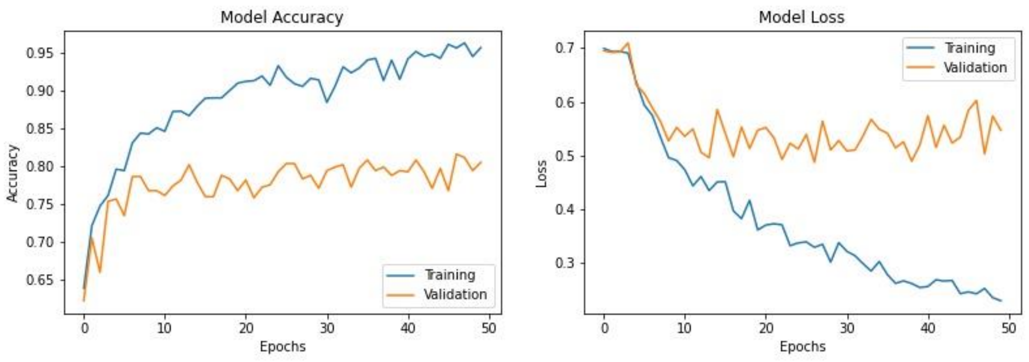


Figure 4.1: Training and Validation Accuracy and Loss Curve for EfficientNet B0

Figure 4.1 shows the accuracy and loss curve for EfficientNet B0. Based on the accuracy on the training and validation, the first 20 epochs show a good increment of accuracy from approximately 65% to more than 90% for training and around 75% - 80% for validation. The final accuracy of the model spiked up until more than 95% in training after 50 epochs have completed. However, the validation accuracy was saturated at ±80% after the overall epoch progress has completed. This might happen due to overfitting when the model is trying to understand our dataset. Based on the loss, the first 10 epochs show a good decrement of loss value from more than ±70% for both training and validation, down to less than 50% for training and nearly reaching 50% for validation. Furthermore, as the epoch sliding through its 50th time, the training loss managed to go down further to lower than 30%. On the other hand, validation loss decreases but not as low as the training loss with final decrement of less than 60% at the 50th epoch.

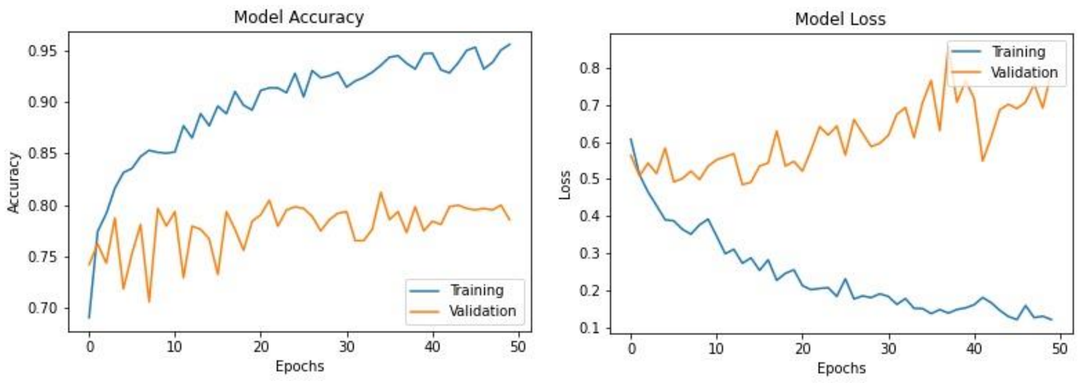


Figure 4.2: Training and Validation Accuracy and Loss Curve for MobileNetV2

MobileNetV2 results shows that its training accuracy managed to go almost 95% and less than 80% for validation accuracy at 50th epoch. Nevertheless, the validation accuracy for MobileNetV2 does not improve as much of difference compared to EfficientNet B0. In terms of loss, MobileNetV2's validation loss increases from less than 60% to almost 80% instead of decreasing. This shows that MobileNetV2 does not perform well as the feature extractor when combined with the RNN classifier. On the other hand, MobileNetV2's training loss went down to almost 10%.

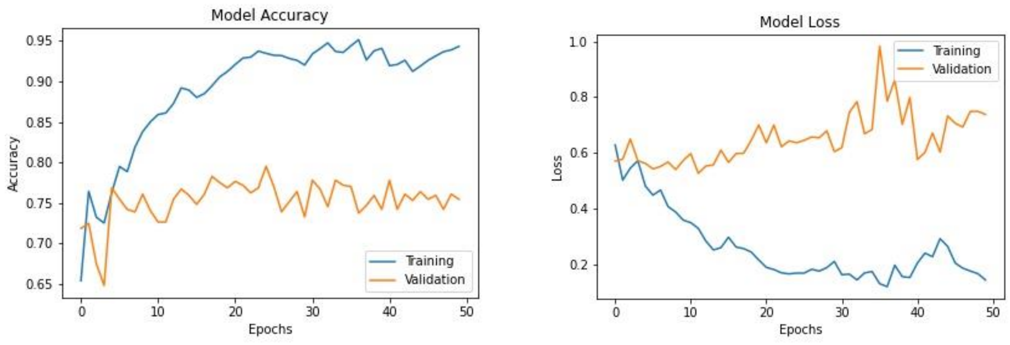


Figure 4.3: Training and Validation Accuracy and Loss Curve for DenseNet-121

DenseNet-121 training accuracy had shown a quite similar results compared to both EfficientNet B0 and MobileNetV2. However, its validation accuracy depicted that the value of it maintain around 65% to ±75% throughout the 50 epochs. In terms of loss, DenseNet-121 has managed to go less than 20% for its training accuracy at 50th epoch. However, the validation loss for DenseNet-121 were not able to get lower than 40%. In fact, the highest value captured for validation loss percentage went for almost 100% between epoch 30 and epoch 40. This has shown that it is not necessarily for a feature extractor to have a bigger depth value, while it cannot provide a decent value during the training with the RNN classifier.

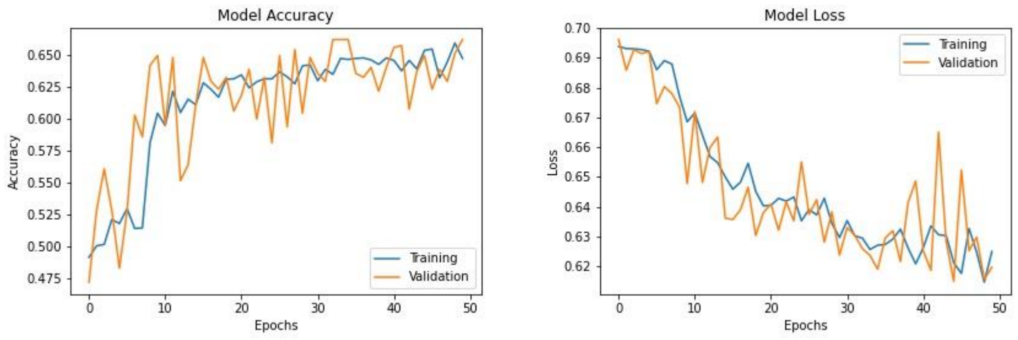


Figure 4.4: Training and Validation Accuracy and Loss Curve for ShuffleNet

As for ShuffleNet, the training and validation accuracy obtained from the experiment shown that it was only able to reach a maximum value of approximately 60%. As for the loss, ShuffleNet were able to show decrement in both training and validation loss value. However, the lowest loss value captured from the experiment of ShuffleNet with RNN classifier was around only 62% which is quite high compared to all other three (3) pre-trained CNN. This happens due to the fact that ShuffleNet has the lowest depth value compared to other models, which constitutes to lower number of trainable parameters available within the ShuffleNet model itself.

4.2 Presentation and Discussion of Result in terms of Confusion Matrix

Another representation of result was obtained using the testing set available from the dataset, which consist of 800 videos came from two (2) classes, violence and non-violence. Confusion Matrix [56] is one of the easiest criteria for determining how precise and right a model is. It is used with classification issues where any class might be the output or the class that the dataset has been categorised into one of the 2 in binary classification and one-to-many in multi-class classification. Although the confusion matrix is not a performance measure in and of itself, most performance measures are built around it and the information it provides.

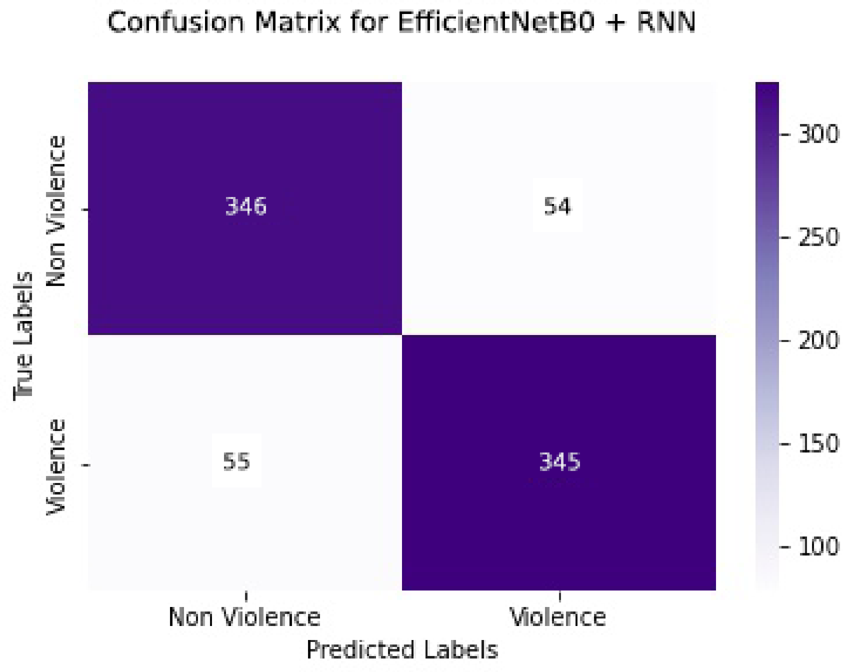


Figure 4.5: Confusion Matrix for EfficientNetB0 + RNN Classifier

800 videos in the testing dataset were passed through the trained RNN classifier after its features have been extracted out by each pre-trained CNN. The

results were obtained as shown in Figure 4.5. There are a total of 691 videos were predicted correctly based on its true classes after the evaluation have been made, comparing between the predicted labels and its true labels stored in another array. This constitutes to 86.375% videos from the testing dataset being predicted correctly in terms of violence and non-violence respectively. Apart from that, there were 109 videos were predicted incorrectly which constitutes to only 13.625% videos were predicted incorrectly after being passed through the RNN classifier. This shows that the combination of EfficientNet B0 and RNN Classifier performs well in performing violence detection in videos.

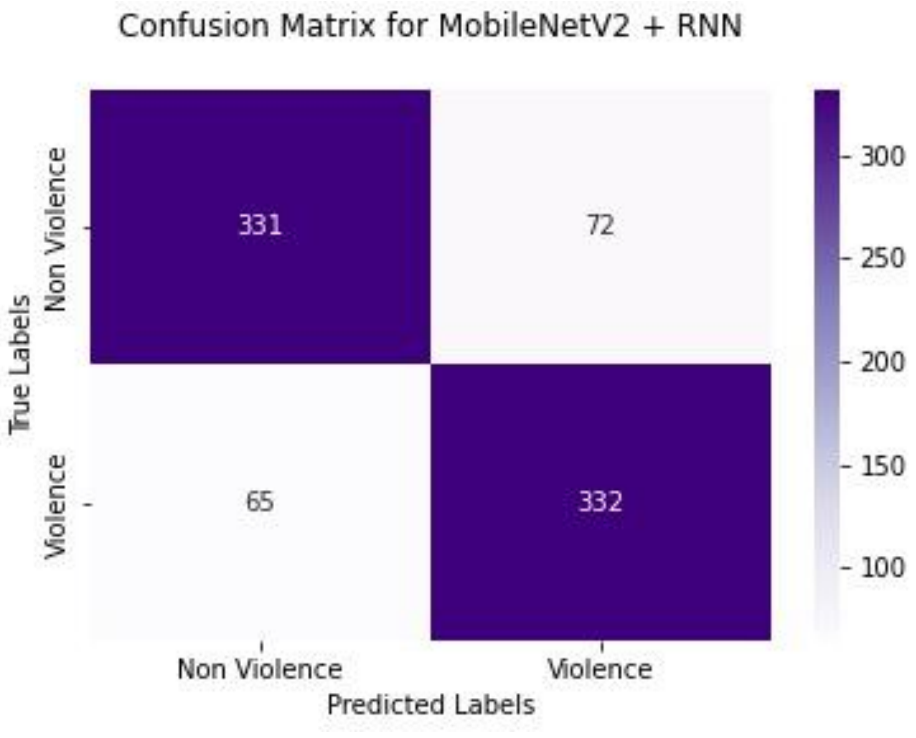


Figure 4.6: Confusion Matrix for MobileNetV2 + RNN Classifier

The confusion matrix of MobileNetV2 was plotted after the feature-extracted videos in the test set has been predicted by the RNN classifier. In comparison to EfficientNet B0, MobileNetV2 has a higher value of 17.125%, whereby it is the rate of falsely predicted videos in the testing dataset. On a side note, MobileNetV2 also has a lower value compared to EfficientNet B0, in terms of correctly predicted videos according to the correct classes, with a rate of 82.875% detection rate.

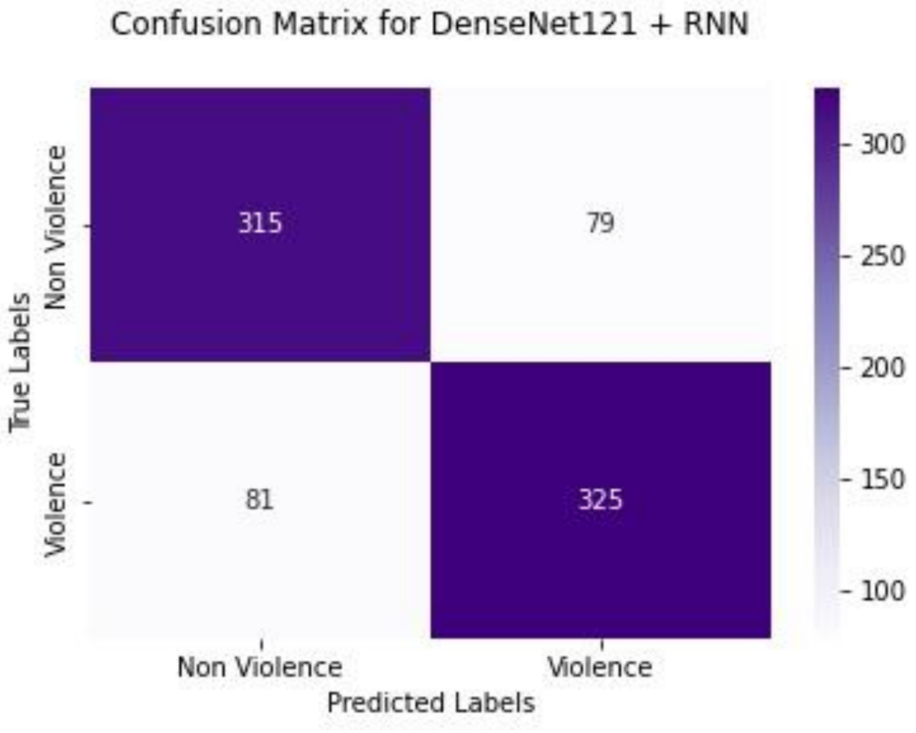


Figure 4.7: Confusion Matrix for DenseNet-121 + RNN Classifier

DenseNet-121 has the highest number of depths, or in other words, the number of layers inside its architecture, were used for feature extraction. The model with the highest number of parameters compared to the other three (3) models did not manage to show the greatest result compared to EfficientNet B0. DenseNet-121 managed to classify only 640 videos out of 800 videos available which constitutes to only 80% videos were predicted according to its correct classes. Besides that, the confusion matrix of DenseNet-121 also shows that it predicted 20% videos inside the test set incorrectly. This means that 81 videos were predicted as non-violence, while it is actually a violence video, and 79 videos were predicted as violence, while it was actually a set of non-violence videos.

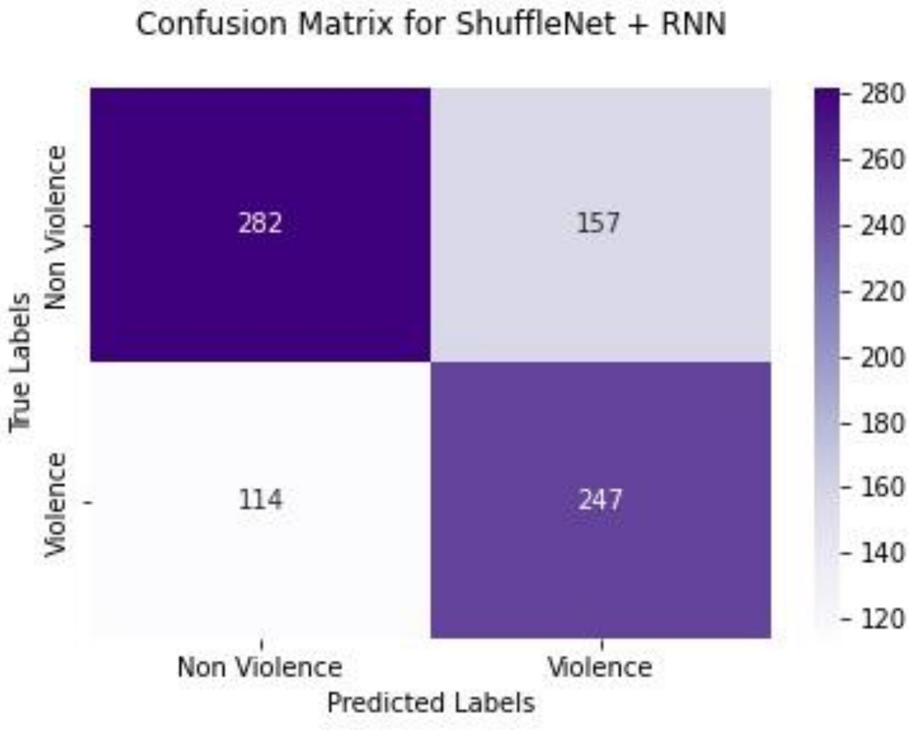


Figure 4.8: Confusion Matrix for ShuffleNet + RNN Classifier

ShuffleNet provides a quite low value in terms of accuracy and high value in accuracy from the experiment of 50 epochs. From figure 4.8, it can be clearly seen that ShuffleNet does not perform well in detecting violence content in videos. A huge number of 157 videos were predicted as violence, while all of them are non-violence. Aside from that, there were only 66.125% videos were predicted correctly using the RNN classifier, after inserting the videos that were feature-extracted by ShuffleNet. Hence, the confusion matrix shown for ShuffleNet + RNN classifier is another proof to assume that it is not a good classifier to be used for violence detection in videos.

4.3 Presentation and Discussion of Result in terms of Classification Report

Classification Report [57] is another way to present the performance of the RNN model based on the predictions made using the testing dataset. This classification report was combined in terms of the metrics used to measure a performance in the classification report. Metrics such as f1-score, recall, accuracy, false positive rate, and precision were used to make up a classification report table, where all its definition can be found in Table 4.1. TP represents True Positive, TN represents True Negatives, FP represents False Positives, FN represents False Negatives. Table 4.2 shows the classification report for violence detection in videos.

Table 4.1: Metrics of Evaluation comprised in a Classification Report

Metrics	Equation	Description of Metrics
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Rate of correctly predicted videos over all videos in the testing dataset.
Recall	$\frac{TP}{TP + FN}$	The proportion of correctly predicted videos in the non-violence classes to the total correctly predicted videos in both violence and non-violence classes.
Precision	$\frac{TP}{TP + FP}$	Rate of correctly predicted non-violence class videos over the total of non-violence videos.
False Positive Rate	$\frac{FP}{FP + TN}$	The measurement proportion of wrongly predicted non-violence videos over the total number of videos that were predicted as violence.
F1-Score	$\frac{2 \times Precision \times Recall}{Precision + Recall}$	The mean of harmonic between the obtained value of recall and precision. Additionally, the closer the value to 1, the better the performance of the classifier is.

Table 4.2: Classification Report on different Feature Extractors (with RNN classifier) for Violence Detection in Videos

Feature Extractors	Accuracy	Recall	Precision	F1-Score	False Positive Rate
MobileNetV2	0.8288	0.8359	0.8213	0.8285	0.1782
EfficientNet B0	0.8638	0.8628	0.8650	0.8639	0.1353
DenseNet121	0.8000	0.7955	0.7995	0.7975	0.1955
ShuffleNet	0.6613	0.7121	0.6424	0.6754	0.3886

Classification Report were obtained from the evaluation of the RNN classifier performance after the testing dataset has been passed through the classifier itself. It projected the values as shown in the Table 4.2 From the table itself, EfficientNet B0 outperforms the other models by getting the highest value in terms of accuracy, recall, precision, f1-score and the smallest value in terms of false positive rate. 86.38% accuracy value were obtained which symbolizes the highest rate obtained for EfficientNet B0 model in predicting the correct video over the testing dataset. A recall value of 86.28% were obtained which signifies that the model was able to predict the said rate for non-violence videos over the total predicted videos that have been made just for both classes. EfficientNet B0 also managed to obtain 86.50% for the ratio of videos that were correctly predicted as non-violence over the total of non-violence videos available from the overall testing dataset. 0.8639 f1-score was obtained to represent the harmonic mean of between the EfficientNet B0’s recall value as well as the precision value, and among all models experimented in this research, EfficientNet B0 has the value that is closest to 1. Finally, the model obtained the lowest rate in terms of the false positive rate with only 13.53%. ShuffleNet, however, obtained the lowest value for the accuracy, recall, precision and f1-score, and the highest value in terms of the false positive rate. This has shown another proof that the combination of ShuffleNet with the RNN Classifier does not suit to be used for violence detection inferencing to avoid any faulty occurred during the implementation stage.

4.4 Real-Time Implementation and IoT Dashboard via Thingspeak

The violence detection was able to be deployed using the best model that have been evaluated which is EfficientNet B0 with the RNN classifier. The implementation was done inside the RaspberryPi 4 Model B 8GB RAM. 150 frames were taken from the camera display with 30 frames per second were set for the USB Web Camera. Hence, the detection will take place for every three (3) seconds during the real-time implementation. In addition to the utilization of Raspberry Pi, three (3) heatsinks as well as Pi fan were used to mitigate the heat produced by the microcontroller to prevent the system from lagged. Figure 4.9, Figure 4.10, and Figure 4.11 shows the real-life setup for violence detection using Raspberry Pi. Figure 4.12 shows the output of the violence prediction on screen.

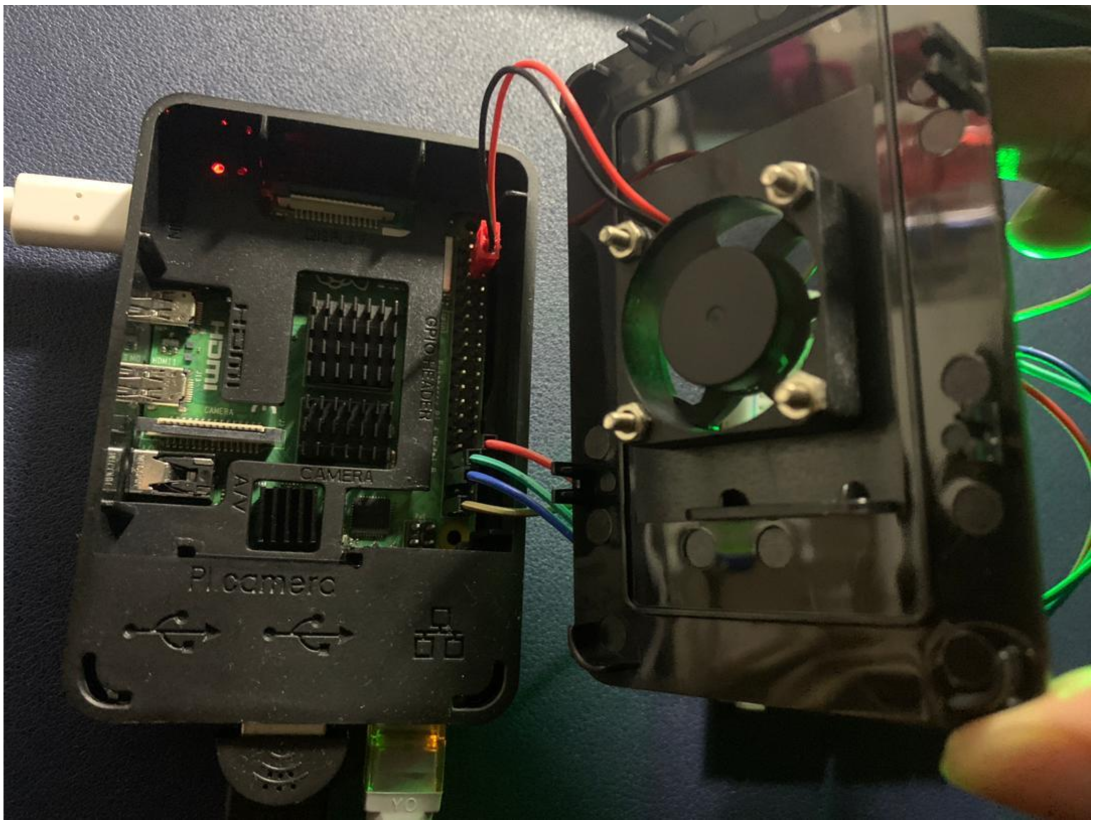


Figure 4.9: Heatsinks and Pi fan attached to the Raspberry Pi

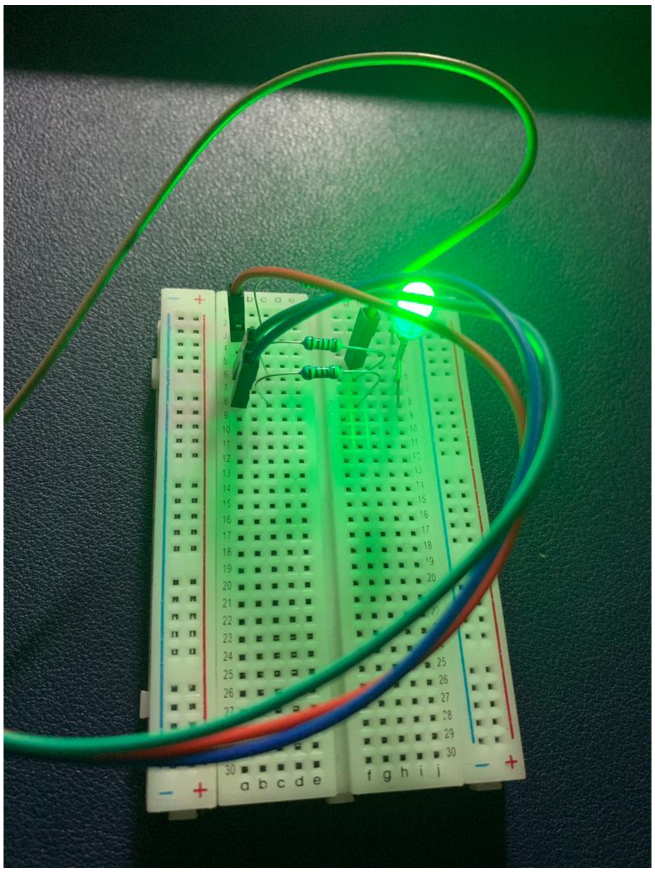


Figure 4.10: RGB LED connected to the Raspberry Pi

Figure 4.11: The USB Web Camera is pointing towards the Violence Videos on played on iPad

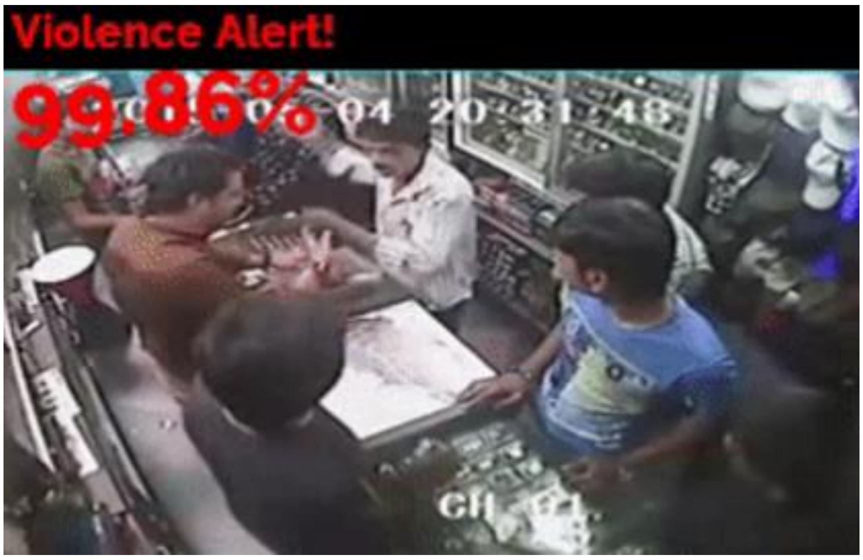
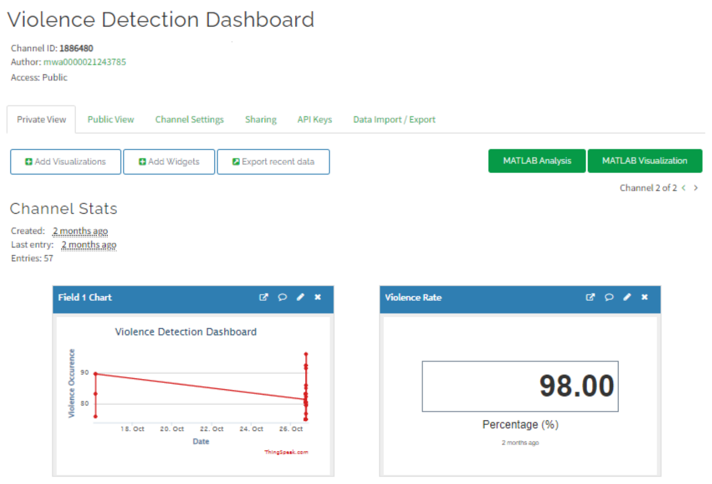


Figure 4.12: Violence Act were detected from the Web Camera

Besides that, the occurrence of violence detected from the Raspberry Pi 4 will be immediately recorded and uploaded to cloud IoT analytics platform via Thingspeak [58]. A platform offering a variety of services specifically designed for developing IoT applications is called Thingspeak. It gives the ability to collect data in real-time, visualise the data in charts, and build plugins and apps for interacting with online services, social networks, and other APIs. Thingspeak cloud IoT analytics platform channel were used to receive and record all the violence detection percentage, timestamp as well as the visualization of the data through graphs.


Violence Detection Dashboard

Channel ID: 1886480
Author: mwa0000021243785
Access: Public

Channel Stats

Created: 2 months ago
Last entry: 2 months ago
Entries: 57

Violence Rate
98.00
Percentage (%)
2 months ago

Figure 4.13: Violence Detection Dashboard via Thingspeak

4.5 Comparison of Result with the Currently Implemented Systems

Table 4.3: Comparison of Result with the Current Implementation

Dataset	**Model Used**	**Accuracy (%)**	**Ref. No. / Proposed Work**
RWF-2000 + RLVS-2000	CNN (Keras Sequential Layers) - LSTM	73.35	[43]
	CNN (MobileNetV2) – RNN	82.88	Proposed Work
	CNN (DenseNet-121) – RNN	80.00	Proposed Work
	CNN (ShuffleNet) – RNN	66.13	Proposed Work (baseline)
	CNN (EfficientNet B0) – RNN	86.38	Proposed Work

Table 4.3 shows the comparison of different model performance in terms of accuracy from the proposed work as well as the currently implemented methodology. Research made by Nouar Aldahoul et. al. has obtained 73.35% accuracy which surpassed the baseline of the proposed work in detecting violent contents using low-cost IoT node. The baseline referred to be the pre-trained model used for the feature extraction process known as ShuffleNet which only managed to reach around 66% for its accuracy. The current proposed work in this research has been made using a few different pre-trained models for feature extraction purposes and hence, different results has been obtained for comparison of previous methodology over the similar dataset utilization.

The current proposed work in this research has made significant contribution in producing better accuracy for detecting violent contents in videos. Even though the previous proposed work has surpassed one of the proposed works in this research, the other three combination of pre-trained models as well as RNN classifier outperformed the previous work by at least 6% higher than the previous work. EfficientNet B0 has the highest accuracy among the three other combinations in the current proposed work and it has also surpassed the previously implemented work by more than 13%, relatively higher and will technically constitutes to a better violence detection in videos.

CHAPTER 5: CONCLUSIONS

This is the last chapter of the Final Year Project (FYP) Report. Several last items will be explained and concluded in this chapter such as the Actual Budget Projection whereby the spendings incurred in purchasing the electronic items will be explained. Besides that, this chapter will also conclude on all works made in completing this research, and this chapter will emphasize on the potential areas for this research to be made expandable or improved further in the future.

5.1 Actual Budget Projection

Several electronic items were purchased to accomplish some parts of the research, mainly on the deployment of the violence detection models on the hardware side. Online vendors were chosen for item purchase to speed up the buying process since it is quite hard to find the electronics items at shops around Cyberjaya area. Table 5.1 summarizes the actual projection of budget used for this research.

Table 5.1: Summary of Actual Projection of Budget

NO	ITEM	QUANTITY	UNIT PRICE (RM)	PRICE (RM)
1	Raspberry Pi 4 Model B Basic Kit (CK-PI4-8G-BS2UK)	1	488.00	488.00
2	Raspberry Pi 4 Heatsink Set (HD-HS-RPI4-3B)	1	5.00	5.00
3	LED 5mm RGB Diffused (DS-LED-5SM-4D-CC)	3	1.00	3.00
TOTAL EXPENSES (RM)				496.00

RM496.00 were used for purchasing from the vendor. The selected vendor was Cytron since it is the most reliable online store that sells electronics materials across Malaysia. The appendix of receipts may be found in the Appendix section in this research.

5.2 Conclusions

In conclusion, this research would have helped a lot of people especially the people inside the community in bringing peace among the neighbourhood community, as well as promoting a healthy mindset towards the children who are currently growing to become a better person day by day. The method of detecting violence situation by pointing out the camera towards the screen would have help a family to use the system at their home by just using the proposed system with a low cost to implement it at their home. Two (2) Sustainable Development Goals (SDG) have a certain correlation towards this research.

16 PEACE, JUSTICE AND STRONG INSTITUTIONS
8 DECENT WORK AND ECONOMIC GROWTH

Figure 5.1: SDG Goals that correlates towards this research

The implementation of this project can be beneficial towards the neighbourhood community, in which correlates to the Goal 16 of SDG, Peace, Justice and Strong Institutions. This system would help the humankind to achieve and sustain peace within the community by improving the reliability of the current security system which indirectly could help in mitigating the number of violent cases that could potentially happen within the community. Additionally, this project would have also help to reduce the need to rely on human patrolling in the crowded places, by just simply implementing the system in the said place, which can effectively reduce the cost to hire manpower for the patrolling services. Moreover, this leads to the SDG 8, Decent Work and Economic Growth since a small amount of people would only be needed during the action-taking stage after a violence situation has been detected. The cooperation with the authorities such as police department would already be enough to further acts or decide the implication towards the person who committed the violence cases. Instead, the investment on manpower could have been diverted more towards the research and development related to artificial intelligence and machine learning which undeniably would have been more effective and more neutral in detecting certain anomaly acts in both private and public areas.

5.3 Areas of Future Research

This research could have been further improved in many ways. Hardware implementation is one of the obvious improvements that can be included to improve this research. The involvement of CUDA-enabled device or a device with a Graphical Processing Unit (GPU) would have helped the training to take place faster so that a faster training could have been done, hence, opening more ways to find a better arbitrarily defined RNN model. Besides that, NVIDIA Jetson Nano Developer Kit can also be used in the future for real-time implementation of violence detection via microcontrollers.

Besides that, the future could implement a way to perform parallel CNN-RNN inferencing to allow parallel detection of violence situation in videos to take place. It would have effectively improved the capability of detection without having to wait for a few milliseconds before the detection of violence situation takes place.

Moreover, the dataset that is currently being used are lack in terms of content coming from movies. Most of the violent situations were coming from the videos captured from the mobile phones, and CCTV. As for future improvements to encourage a better detection of violent situation in movies, a fighting scene from movies, films, or dramas with different camera angles involved would have helped the researchers to further investigate the actions based on the movies and films.

REFERENCES

[1] Allan Koay, "Cutting for Change", *TheStar Online*, May 2007, Available: https://web.archive.org/web/20070930153851/http://starecentral.com/news/story.asp?file=%2F2007%2F5%2F14%2Fmovies%2F17639243&sec=movies [Accessed February 2022]

[2] American Academy of Child and Adolescent Psychiatry, "Movies, Media, and Children", *Facts for Families,* Available: https://www.aacap.org/AACAP/Families_and_Youth/Facts_for_Families/FFFGuide/Children-And-Movies-090.aspx [Accessed February 2022]

[3] NVIDIA Corporation, "Optimizing Linear/Fully-Connected Layers", *NVIDIA Docs,* Available: https://docs.nvidia.com/deeplearning/performance/pdf/Optimizing-Linear-Fully-Connected-Layers-User-Guide.pdf [Accessed November 2022]

[4] Aman Kharwal, "What are Pre Trained Models?", *The Clever Programmer,* March 2021, Available: https://thecleverprogrammer.com/2021/03/27/what-are-pre-trained-models/ [Accessed November 2022]

[5] Mathworks, "Feature Extraction", *Discover How to Solve Your Computational Problem,* Available: https://www.mathworks.com/discovery.html [Accessed November 2022]

[6] Olmos, R., Tabik, S., and Herrera, F.: Automatic handgun detection alarm in videos using deep learning. *Neurocomputing*, 275, 66-72 (2018).

[7] Li, X., and Shi, Y.: Computer vision imaging based on artificial intelligence. in Proc. *Int. Conf. Virtual Reality Intell. Syst (ICVRIS)*, 22-25(2018).

[8] Deniz, O., Serrano, I., Bueno, G., and Kim, T.-K.: Fast violence detection in video. In Proc. *Int. Conf. Comput. Vis. Theory Appl.* (VISAPP) (2), 478-485 (2014).

[9] Ribeiro, P. C., Audigier, R., and Pham, Q. C.: RIMOC, a feature to discriminate unstructured motions: Application to violence detection for video surveillance. *Comput. Vis. Image Understand*. 144, 121-143 (2016).

[10] Arceda. V. E. M., Fabián, K. M. F., Laura, P. C. L., and Cáceres J. C. G.: Fast face detection in violent video scenes. *Electron. Notes Theor. Comput. Sci.*, 329, 5-26 (2016).

[11] Xie, J., Yan, W., Mu, C., Liu, T., Li, P., and Yan, S.: Recognizing violent activity without decoding video streams. *Optik*, 127(2), 795-801 (2016).

[12] Fu, E. Y., Va Leong, H., Ngai, G., and Chan, S.: Automatic fight detection in surveillance videos. in Proc. *14th Int. Conf. Adv. Mobile Comput. Multi Media*, 225-234 (2016)

[13] Nar, R., Singal, A., and Kumar, P.: Abnormal activity detection for bank ATM surveillance. in Proc. *Int. Conf. Adv. Comput., Commun. Inform.* (ICACCI), 2042-2046 (2016).

[14] Chaudhary, S., Khan, M. A., and Bhatnagar, C. Multiple anomalous activity detection in videos. *Procedia Comput. Sci.* (125) , 336-345 (2018).

[15] Senst, T., Eiselein, V., Kuhn, A., and Sikora, T.: Crowd violence detection using global motion-compensated Lagrangian features and scale-sensitive video-level representation. *IEEE Trans. Inf. Forensics Security*, 12(12), 2945-2956 (2017).

[16] Fu, E. Y., Huang, M. X., Va Leong, H., and Ngai, G.: Cross-species learning: A low-cost approach to learning human fight from animal fight. in Proc. *26th ACM Int. Conf. Multimedia*, 320-327 (2018).

[17] Hassner, T., Itcher, Y., and Kliper-Gross, O.: Violent fows: Real-time detection of violent crowd behavior. in Proc. *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, 1-6 (2012).

[18] Li, X., Huo, Y., Jin, Q., and Xu, J.: Detecting violence in video using subclasses. in Proc. *24th ACM Int. Conf. Multimedia*, Oct. 586-590 (2016).

[19] Bilinski P., and Bremond, F.: Human violence recognition and detection in surveillance videos. in Proc. *13th IEEE Int. Conf. Adv. Video Signal Based Surveill* (AVSS), 30-36 (2016).

[20] Gao, Y., Liu, H., Sun, X., Wang, C., and Liu, Y.: Violence detection using oriented violent Fows. *Image Vis. Comput.*, 48, 37-41 (2016).

[21] Dhiman, C., and Vishwakarma, D., K.: High dimensional abnormal human activity recognition using histogram oriented gradients and Zernike moments. in Proc. *IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, 1-4 (2017).

[22] Al-Nawashi, M., Al-Hazaimeh, O. M., and Saraee, M.: A novel framework for intelligent surveillance system based on abnormal human activity detection in academic environments. *Neural Comput. Appl.*, 28(1), 565-572 (2017).

[23] Mishra A. A., and Srinivasa, G.: Automated detection of fighting styles using localized action features. in Proc. *2nd Int. Conf. Inventive Syst. Control (ICISC)*, 1385-1389 (2018).

[24] Peixoto, B. M., Avila, S., Dias, Z., and Rocha, A.: Breaking down violence: A deep-learning strategy to model and classify violence in videos. in Proc. *13th Int. Conf. Availability, Rel. Secur.*, 50, (2018).

[25] Song, D., Kim, C., and Park, S-K.: A multi-temporal framework for high level activity analysis: Violent event detection in visual surveillance. *Inf. Sci.*, 447, 83-103 (2018).

[26] Xia, Q., Zhang, P., Wang, J., Tian, M., and Fei, C.: Real time violence detection based on deep spatio-temporal features. in Proc. *Chin. Conf. Biometric Recognit.*, 57-165 (2018).

[27] Agrawal, T., Kumar, A., and Saraswat, S. K.: Comparative analysis of convolutional codes based on ML decoding. in Proc. *2nd Int. Conf. Commun. Control Intell. Syst. (CCIS)*, 41-45 (2016).

[28] Ding, C., Fan, S., Zhu, M., Feng, W., and Jia, B.: Violence detection in video by using 3D convolutional neural networks. in Proc. *Int. Symp. Visual Comput.*, 551-558 (2014).

[29] Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J.: NetVLAD: CNN Architecture for weakly supervised place recognition. in Proc. *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 5297-5307 (2016).

[30] Mu, G., Cao, H., and Jin, Q.: Violent scene detection using convolutional neural networks and deep audio features. in Proc. *Chin. Conf. Pattern Recognit.*, 451-463 (2016).

[31] Sudhakaran, S., and Lanz, O.: Learning to detect violent videos using convolutional long short-term memory. in Proc. *14th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, 1-6 (2017).

[32] Meng, Z., Yuan, J., and Li, Z.: Trajectory-pooled deep convolutional networks for violence detection in videos. in Proc. *Int. Conf. Comput. Vis. Syst.*, 437-447 (2017).

[33] I. S. Gracia, O. Deniz, J. L. E. Aranda, G. Bueno.: Fight Recognition in Video Using Hough Forests and 2D Convolutional Neural Network. In: June 2018, IEEE Transactions on Image Processing PP(99):1-1

[34] Fenil, E., Manogaran, G., Vivekananda, G. N., Thanjaivadivel, T., Jeeva, S., and Ahilan, A.: Real time violence detection framework for football stadium comprising of big data analysis and deep learning through bidirectional LSTM. *Computer Networks*, 151, 191-200 (2019).

[35] Yu, Y., Si, X., Hu, C., and Zhang, J.: A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation*, 31(7), 1235-1270 (2019).

[36] Hanson, A., Pnvr, K., Krishnagopal, S., and Davis, L.: Bidirectional convolutional LSTM for the detection of violence in videos. In Proceedings of the *European Conference on Computer Vision (ECCV) Workshops*, (2018).

[37] E. B. Nievas et al., "Movies fight detection dataset", *Computer Analysis of Images and Patterns*, pp. 332-339, 2011.

[38] E. B. Nievas et al., "Hockey fight detection dataset", *Computer Analysis of Images and Patterns*, pp. 332-339, 2011.

[39] T. Hassner, Y. Itcher and O. Kliper-Gross, "Violent flows: Real-time detection of violent crowd behavior", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1-6, 2012.

[40] M M. Cheng, K. Cai and M. Li, "RWF-2000: An Open Large Scale Video Database for Violence Detection," *25th International Conference on Pattern Recognition (ICPR),* 2021, pp. 4183-4190, doi: 10.1109/ICPR48806.2021.9412502.

[41] M. M. Soliman, M. H. Kamal, M. A. El-Massih Nashed, Y. M. Mostafa, B. S. Chawky and D. Khattab, "Violence Recognition from Videos using Deep Learning Techniques," *Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2019, pp. 80-85, doi: 10.1109/ICICIS46948.2019.9014714.

[42] Ravina Dable, "Violence Detection from Videos Captured by CCTV", *Medium.com,* 26th October 2020.

[43] Nouar AlDahoul, "Convolutional Neural Network - Long Short-Term Memory based IOT Node for Violence Detection", *IEEE International Conference on Artificial Intelligence in Engineering and Technology*, 28th October 2021, Available: IEEEXplore.

[44] Ademir Rafael Marques Guedes, "Real-Time Violence Detection in Videos using Dynamic Images", *2020 XLVI Latin American Computing Conference,* 28th June 2021, Available: IEEEXplore.

[45] Rasoul Banaeeyan, "Automated Nudity Recognition using Very Deep Residual Learning Network", *International Journal of Recent Technology and Engineering,* October 2019, Available: IEEEXplore.

[46] Mi Young Lee, "Cover the Violence: A Novel Deep-Learning-Based Approach Towards Violence-Detection in Movies", *Dept. of Software, Sejong University*, 18 November 2019, Available: Molecular Diversity Preservation International.

[47] Benedikt Droste, "Google Colab Pro+: Is it worth $49.99?", *Towards Data Science,* August 2021

[48] Microsoft, "Visual Studio Code Frequently Asked Questions", *Visual Studio Code* [Accessed December 2022]

[49] Mingxing Tan, Quoc V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", *arXiv.org*, 2020, Retrieved from https://arxiv.org/pdf/1905.11946v5.pdf, in November 2022.

[50] Mark Sandler, Andrew Howard, "MobileNetV2: The Next Generation of On-Device Computer Vision Networks", *Google Research*, April 2018, Retrieved from https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html, in November 2022.

[51] G. Huang, Z. Liu, van, and Weinberger, Kilian Q, "Densely Connected Convolutional Networks," *arXiv.org*, 2016, Retrieved from https://arxiv.org/abs/1608.06993, in November 2022.

[52] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices", *Cornell University*, July 2017, Retrieved from https://arxiv.org/abs/1707.01083, in November 2022.

[53] IBM, "What is Recurrent Neural Networks?", *IBM*, Retrieved from https://www.ibm.com/topics/ [Accessed December 2022]

[54] Huma Abidi, "AI inference acceleration on CPUs", *Intel*, December 2021, Retrieved from https://venturebeat.com/ [Accessed December 2022]

[55] Red Hat, "What is a Raspberry Pi?", *Red Hat*, Retrieved from https://opensource.com/resources/ [Accessed December 2022]

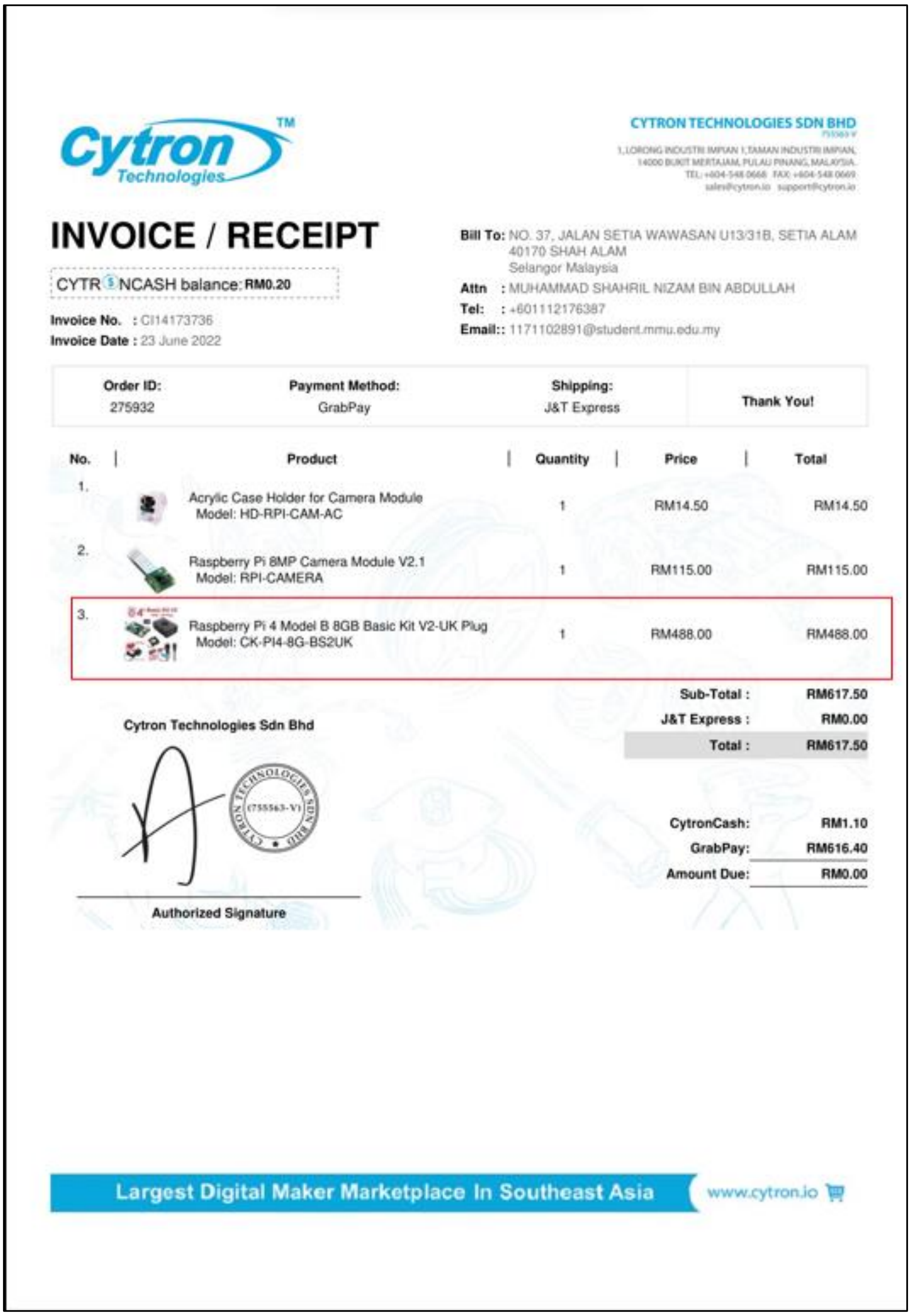
[56] Knowledgehut, "Confusion matrix in machine learning", *Knowledgehut*, Retrieved from https://www.knowledgehut.com/tutorials/ [Accessed December 2022]

[57] Scikit learn, "Metrics and scoring: quantifying the quality of predictions", *Scikit learn*, Retrieved from https://scikit-learn.org/ [Accessed December 2022]

[58] Balaji Srinivasan, "Internet of Things", *Ubuntu*, [Accessed December 2022]

APPENDIX A

- Expenses incurred for hardware purchasing.



Cytron Technologies™

CYTRON TECHNOLOGIES SDN BHD
755563-V
1, LORONG INDUSTRI IMPIAN 1, TAMAN INDUSTRI IMPIAN,
14000 BUKIT MERTAJAM, PULAU PINANG, MALAYSIA.
TEL: +604-548 0668 FAX: +604-548 0669
sales@cytron.io support@cytron.io

INVOICE / RECEIPT

CYTRONCASH balance: **RM0.20**

Invoice No. : CI14173736
Invoice Date : 23 June 2022

Bill To: NO. 37, JALAN SETIA WAWASAN U13/31B, SETIA ALAM
40170 SHAH ALAM
Selangor Malaysia
Attn : MUHAMMAD SHAHRIL NIZAM BIN ABDULLAH
Tel: : +601112176387
Email:: 1171102891@student.mmu.edu.my

Order ID:	Payment Method:	Shipping:	
275932	GrabPay	J&T Express	Thank You!

No.	Product	Quantity	Price	Total
1.	Acrylic Case Holder for Camera Module Model: HD-RPI-CAM-AC	1	RM14.50	RM14.50
2.	Raspberry Pi 8MP Camera Module V2.1 Model: RPI-CAMERA	1	RM115.00	RM115.00
3.	Raspberry Pi 4 Model B 8GB Basic Kit V2-UK Plug Model: CK-PI4-8G-BS2UK	1	RM488.00	RM488.00

Sub-Total :	**RM617.50**
J&T Express :	**RM0.00**
Total :	**RM617.50**
CytronCash:	**RM1.10**
GrabPay:	**RM616.40**
Amount Due:	**RM0.00**

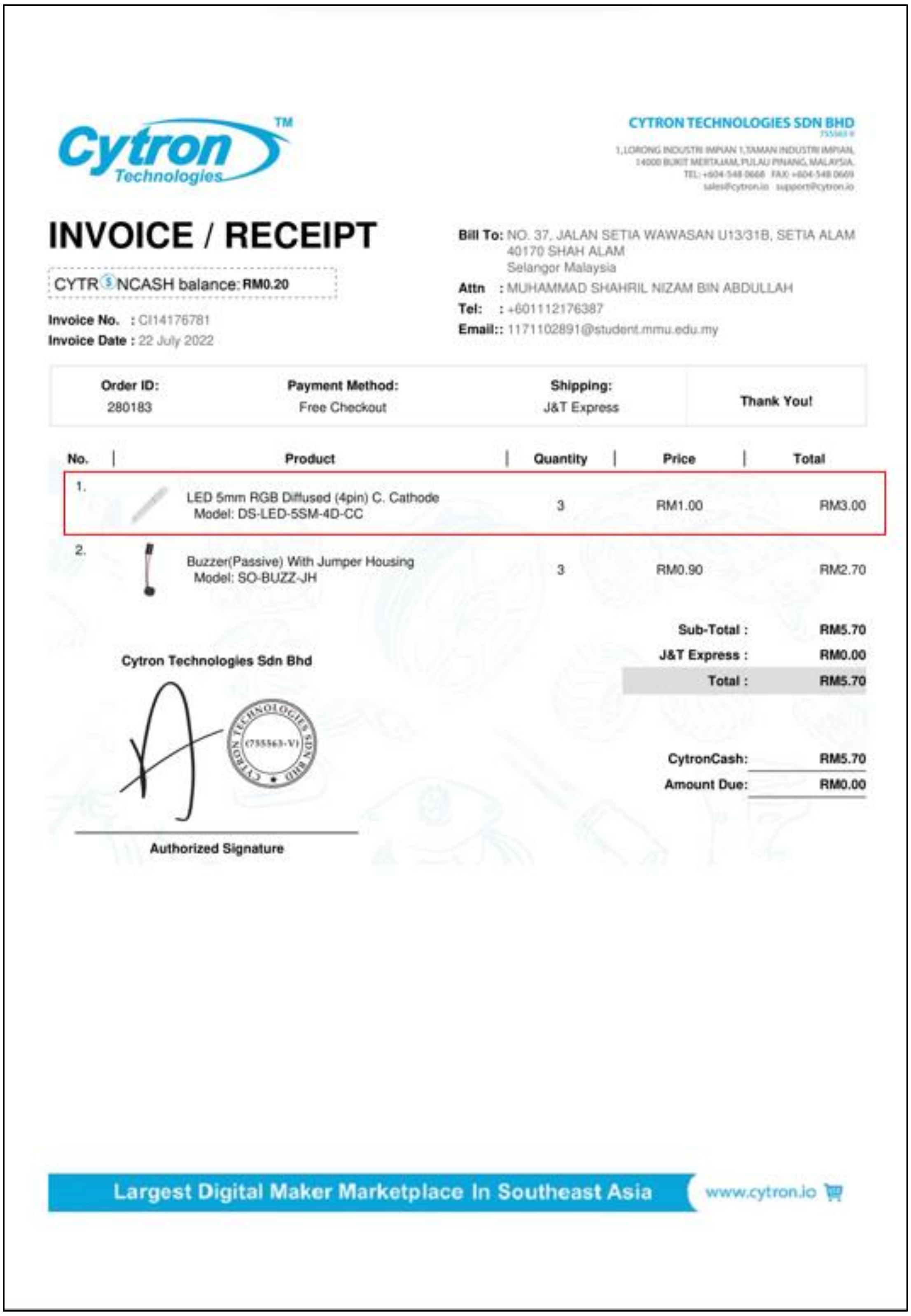
Cytron Technologies Sdn Bhd

[illegible]

CYTRON TECHNOLOGIES SDN BHD (755563-V)

Authorized Signature

Largest Digital Maker Marketplace In Southeast Asia www.cytron.io



Cytron Technologies™

CYTRON TECHNOLOGIES SDN BHD
755563-V
1, LORONG INDUSTRI IMPIAN 1, TAMAN INDUSTRI IMPIAN,
14000 BUKIT MERTAJAM, PULAU PINANG, MALAYSIA.
TEL: +604-548 0668 FAX: +604-548 0669
sales@cytron.io support@cytron.io

INVOICE / RECEIPT

CYTRONCASH balance: **RM0.20**

Invoice No. : CI14176781
Invoice Date : 22 July 2022

Bill To: NO. 37, JALAN SETIA WAWASAN U13/31B, SETIA ALAM
40170 SHAH ALAM
Selangor Malaysia
Attn : MUHAMMAD SHAHRIL NIZAM BIN ABDULLAH
Tel: : +601112176387
Email:: 1171102891@student.mmu.edu.my

Order ID:	**Payment Method:**	**Shipping:**	
280183	Free Checkout	J&T Express	**Thank You!**

No.	**Product**	**Quantity**	**Price**	**Total**
1.	LED 5mm RGB Diffused (4pin) C. Cathode Model: DS-LED-5SM-4D-CC	3	RM1.00	RM3.00
2.	Buzzer(Passive) With Jumper Housing Model: SO-BUZZ-JH	3	RM0.90	RM2.70

Sub-Total :	**RM5.70**
J&T Express :	**RM0.00**
Total :	**RM5.70**
CytronCash:	**RM5.70**
Amount Due:	**RM0.00**

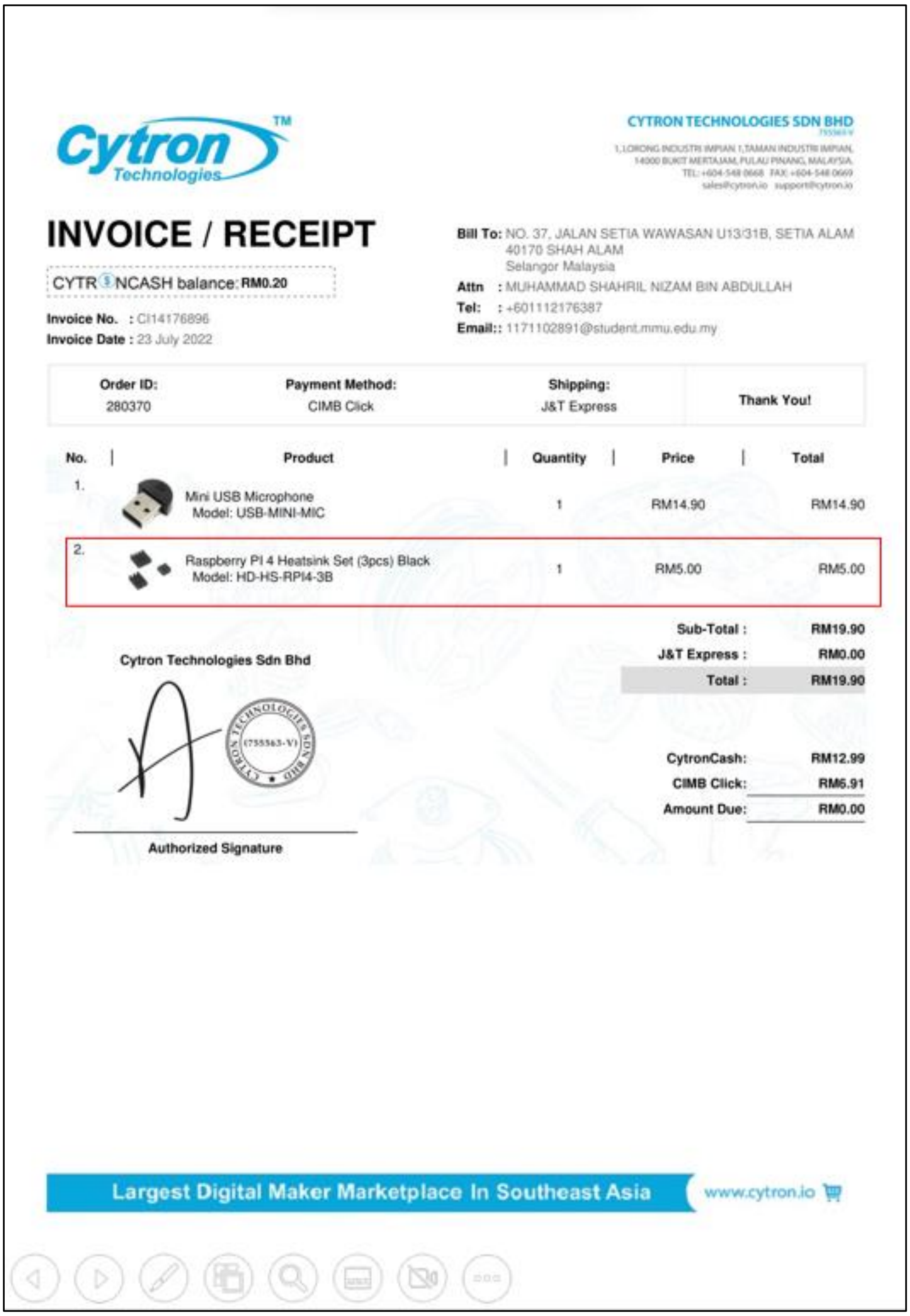
Cytron Technologies Sdn Bhd

[illegible]

CYTRON TECHNOLOGIES SDN BHD (755563-V)

Authorized Signature

Largest Digital Maker Marketplace In Southeast Asia www.cytron.io



CYTRON TECHNOLOGIES SDN BHD
755563-V
1, LORONG INDUSTRI IMPIAN 1, TAMAN INDUSTRI IMPIAN,
14000 BUKIT MERTAJAM, PULAU PINANG, MALAYSIA.
TEL: +604-548 0668 FAX: +604-548 0669
sales@cytron.io support@cytron.io

INVOICE / RECEIPT

CYTRONCASH balance: **RM0.20**

Invoice No. : CI14176896
Invoice Date : 23 July 2022

Bill To: NO. 37, JALAN SETIA WAWASAN U13/31B, SETIA ALAM
40170 SHAH ALAM
Selangor Malaysia
Attn : MUHAMMAD SHAHRIL NIZAM BIN ABDULLAH
Tel: : +601112176387
Email:: 1171102891@student.mmu.edu.my

Order ID:	Payment Method:	Shipping:	
280370	CIMB Click	J&T Express	Thank You!

No.	Product	Quantity	Price	Total
1.	Mini USB Microphone Model: USB-MINI-MIC	1	RM14.90	RM14.90
2.	Raspberry PI 4 Heatsink Set (3pcs) Black Model: HD-HS-RPI4-3B	1	RM5.00	RM5.00

Sub-Total :	RM19.90
J&T Express :	RM0.00
Total :	RM19.90
CytronCash:	RM12.99
CIMB Click:	RM6.91
Amount Due:	RM0.00

Cytron Technologies Sdn Bhd

[illegible]

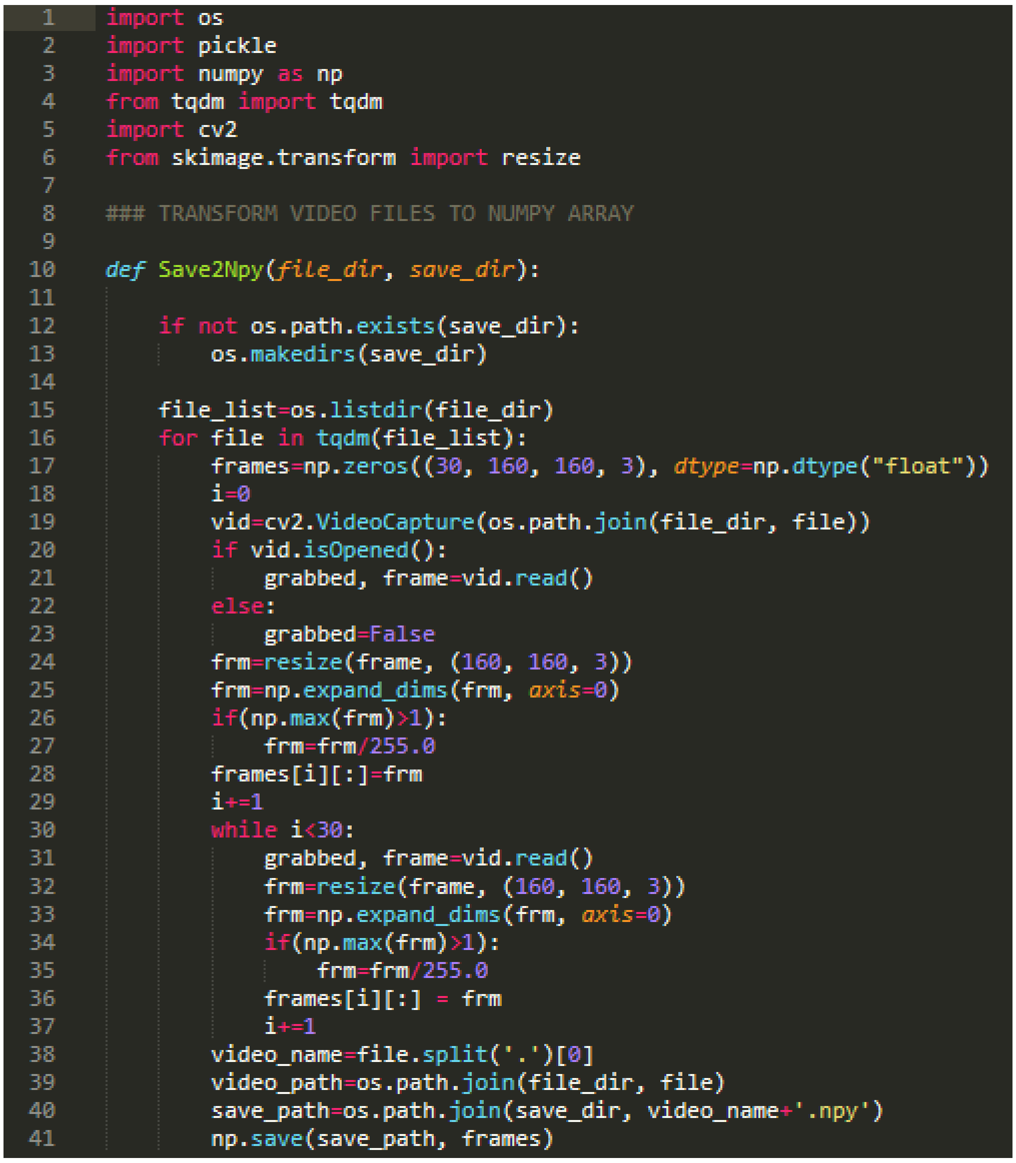
CYTRON TECHNOLOGIES SDN BHD (755563-V)

Authorized Signature

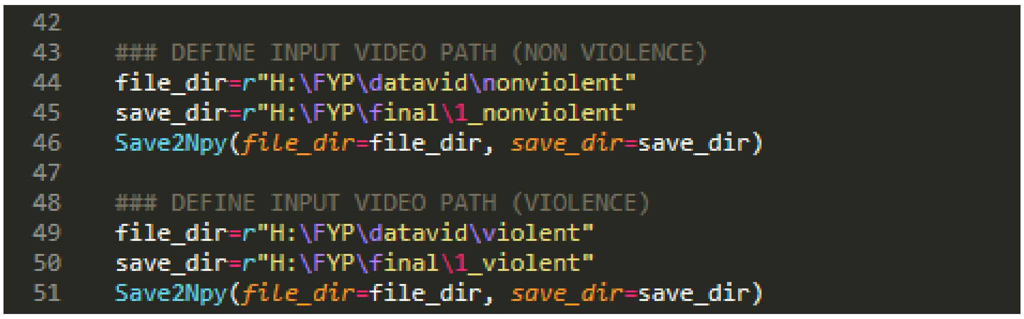
Largest Digital Maker Marketplace In Southeast Asia www.cytron.io

APPENDIX B

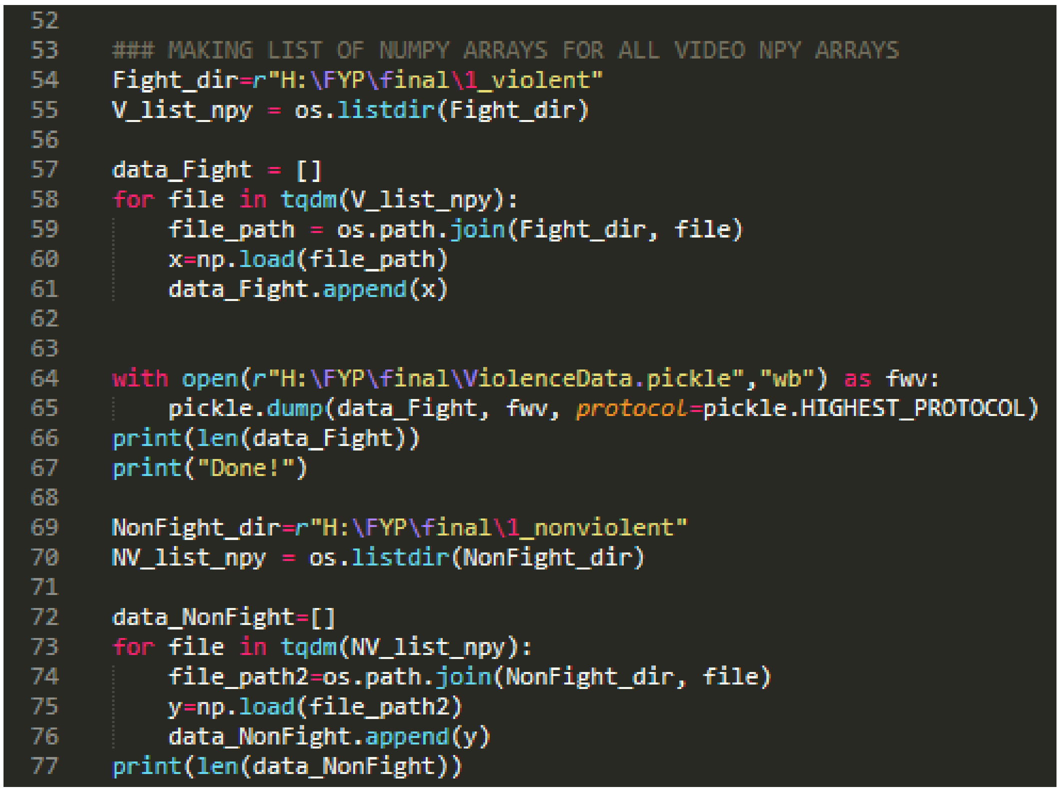
- Conversion of Videos to NumPy Array



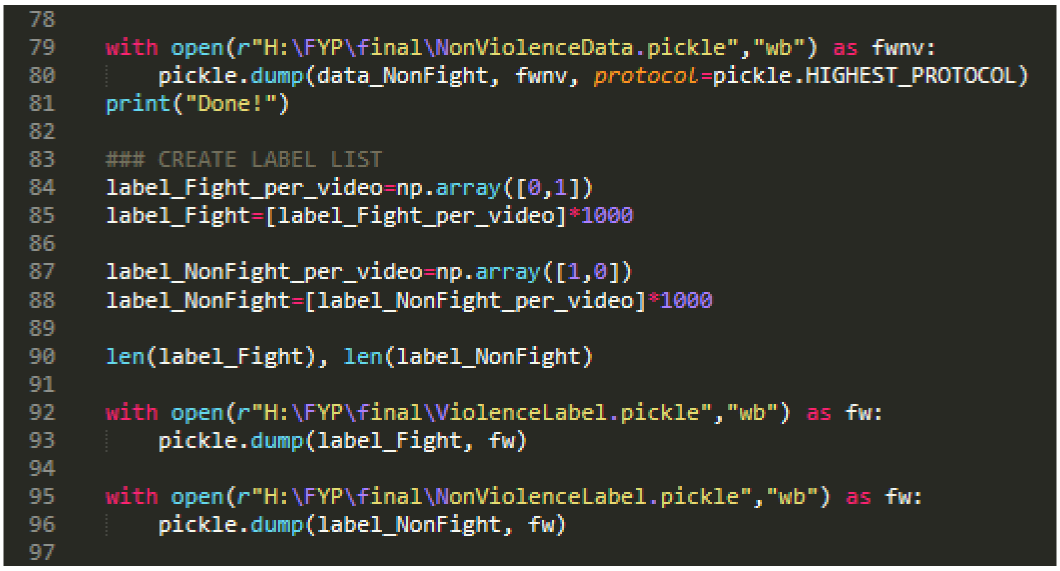
```python
import os
import pickle
import numpy as np
from tqdm import tqdm
import cv2
from skimage.transform import resize

### TRANSFORM VIDEO FILES TO NUMPY ARRAY

def Save2Npy(file_dir, save_dir):

    if not os.path.exists(save_dir):
        os.makedirs(save_dir)

    file_list=os.listdir(file_dir)
    for file in tqdm(file_list):
        frames=np.zeros((30, 160, 160, 3), dtype=np.dtype("float"))
        i=0
        vid=cv2.VideoCapture(os.path.join(file_dir, file))
        if vid.isOpened():
            grabbed, frame=vid.read()
        else:
            grabbed=False
        frm=resize(frame, (160, 160, 3))
        frm=np.expand_dims(frm, axis=0)
        if(np.max(frm)>1):
            frm=frm/255.0
        frames[i][:]=frm
        i+=1
        while i<30:
            grabbed, frame=vid.read()
            frm=resize(frame, (160, 160, 3))
            frm=np.expand_dims(frm, axis=0)
            if(np.max(frm)>1):
                frm=frm/255.0
            frames[i][:] = frm
            i+=1
        video_name=file.split('.')[0]
        video_path=os.path.join(file_dir, file)
        save_path=os.path.join(save_dir, video_name+'.npy')
        np.save(save_path, frames)
```



```python

### DEFINE INPUT VIDEO PATH (NON VIOLENCE)
file_dir=r"H:\FYP\datavid\nonviolent"
save_dir=r"H:\FYP\final\1_nonviolent"
Save2Npy(file_dir=file_dir, save_dir=save_dir)

### DEFINE INPUT VIDEO PATH (VIOLENCE)
file_dir=r"H:\FYP\datavid\violent"
save_dir=r"H:\FYP\final\1_violent"
Save2Npy(file_dir=file_dir, save_dir=save_dir)
```

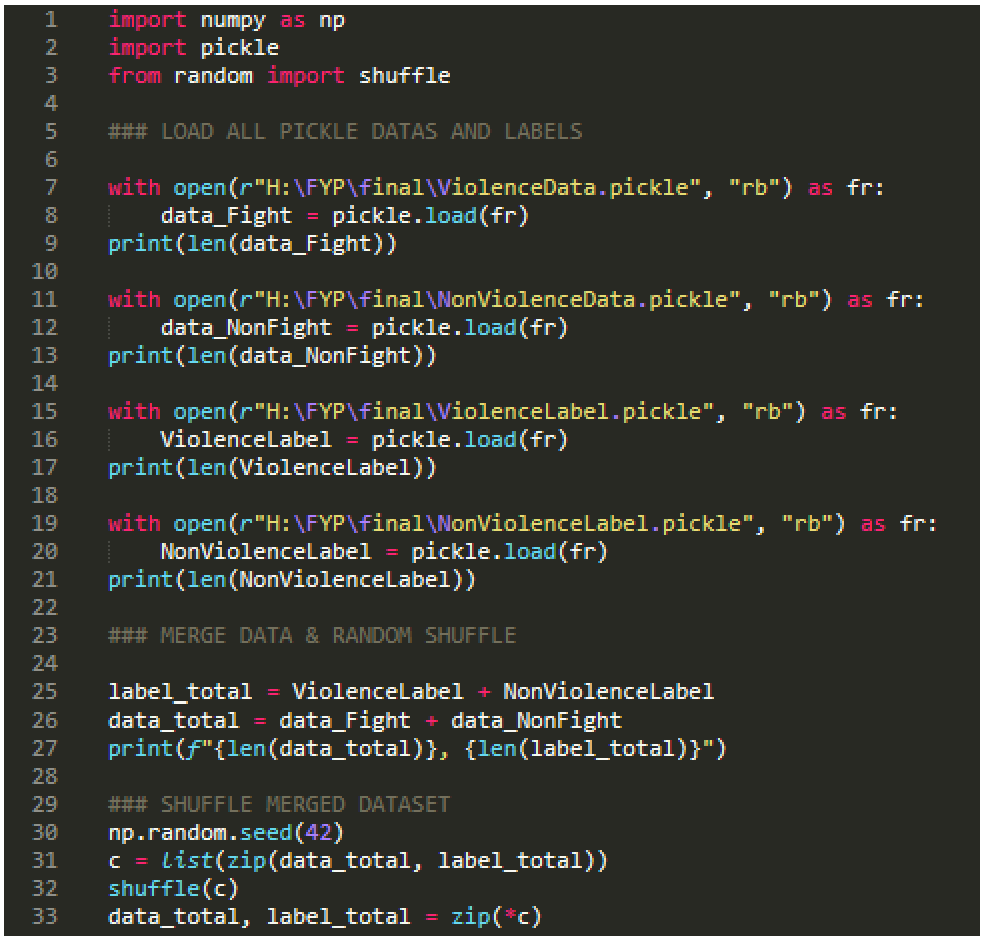


```python

### MAKING LIST OF NUMPY ARRAYS FOR ALL VIDEO NPY ARRAYS
Fight_dir=r"H:\FYP\final\1_violent"
V_list_npy = os.listdir(Fight_dir)

data_Fight = []
for file in tqdm(V_list_npy):
    file_path = os.path.join(Fight_dir, file)
    x=np.load(file_path)
    data_Fight.append(x)


with open(r"H:\FYP\final\ViolenceData.pickle","wb") as fwv:
    pickle.dump(data_Fight, fwv, protocol=pickle.HIGHEST_PROTOCOL)
print(len(data_Fight))
print("Done!")

NonFight_dir=r"H:\FYP\final\1_nonviolent"
NV_list_npy = os.listdir(NonFight_dir)

data_NonFight=[]
for file in tqdm(NV_list_npy):
    file_path2=os.path.join(NonFight_dir, file)
    y=np.load(file_path2)
    data_NonFight.append(y)
print(len(data_NonFight))
```

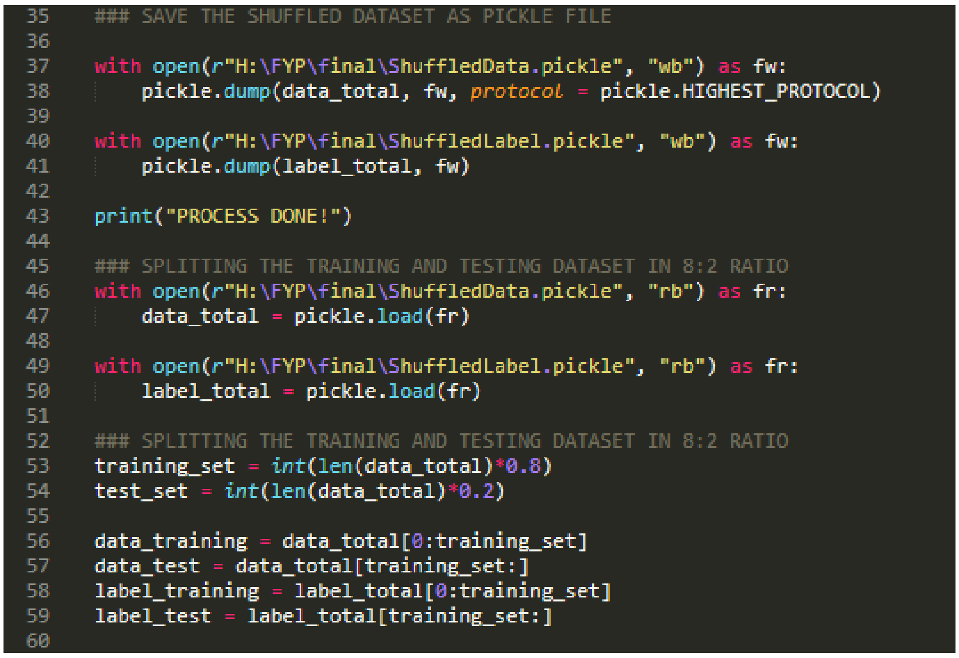


```python

with open(r"H:\FYP\final\NonViolenceData.pickle","wb") as fwnv:
    pickle.dump(data_NonFight, fwnv, protocol=pickle.HIGHEST_PROTOCOL)
print("Done!")

### CREATE LABEL LIST
label_Fight_per_video=np.array([0,1])
label_Fight=[label_Fight_per_video]*1000

label_NonFight_per_video=np.array([1,0])
label_NonFight=[label_NonFight_per_video]*1000

len(label_Fight), len(label_NonFight)

with open(r"H:\FYP\final\ViolenceLabel.pickle","wb") as fw:
    pickle.dump(label_Fight, fw)

with open(r"H:\FYP\final\NonViolenceLabel.pickle","wb") as fw:
    pickle.dump(label_NonFight, fw)

```

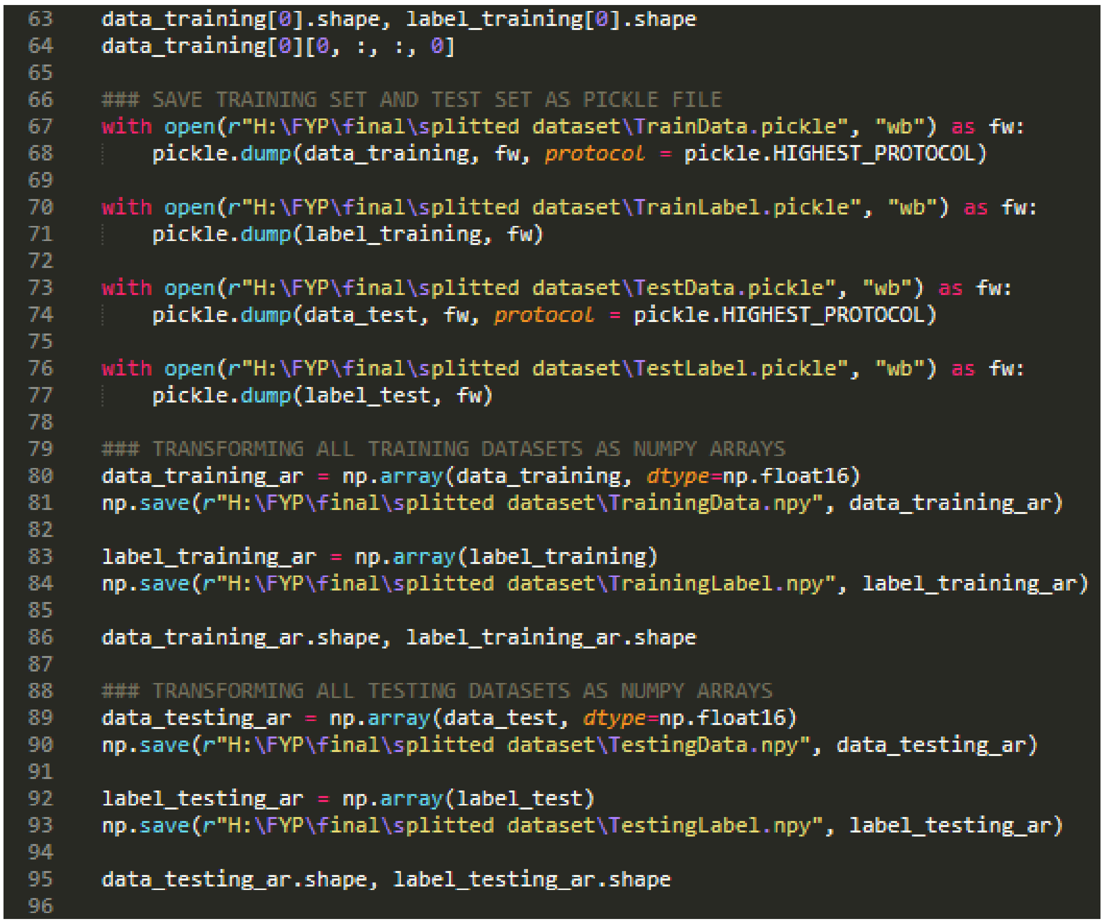
- Train Test Split



```python
import numpy as np
import pickle
from random import shuffle

### LOAD ALL PICKLE DATAS AND LABELS

with open(r"H:\FYP\final\ViolenceData.pickle", "rb") as fr:
    data_Fight = pickle.load(fr)
print(len(data_Fight))

with open(r"H:\FYP\final\NonViolenceData.pickle", "rb") as fr:
    data_NonFight = pickle.load(fr)
print(len(data_NonFight))

with open(r"H:\FYP\final\ViolenceLabel.pickle", "rb") as fr:
    ViolenceLabel = pickle.load(fr)
print(len(ViolenceLabel))

with open(r"H:\FYP\final\NonViolenceLabel.pickle", "rb") as fr:
    NonViolenceLabel = pickle.load(fr)
print(len(NonViolenceLabel))

### MERGE DATA & RANDOM SHUFFLE

label_total = ViolenceLabel + NonViolenceLabel
data_total = data_Fight + data_NonFight
print(f"{len(data_total)}, {len(label_total)}")

### SHUFFLE MERGED DATASET
np.random.seed(42)
c = list(zip(data_total, label_total))
shuffle(c)
data_total, label_total = zip(*c)
```

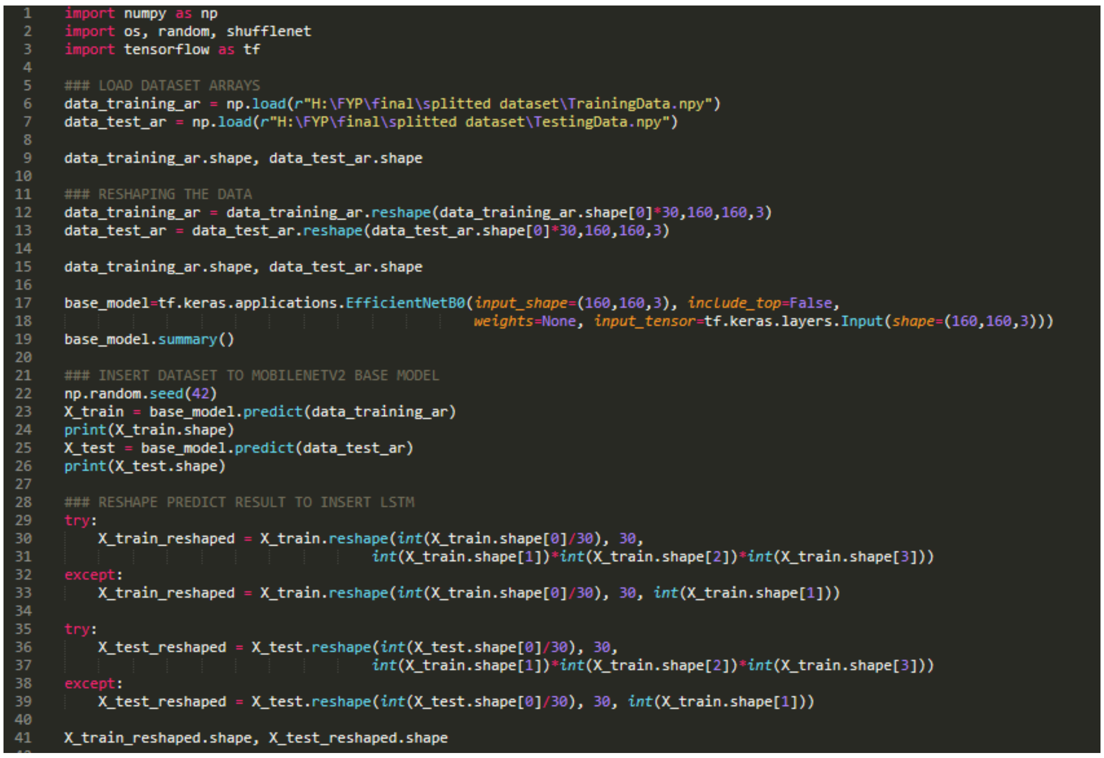


```python
### SAVE THE SHUFFLED DATASET AS PICKLE FILE

with open(r"H:\FYP\final\ShuffledData.pickle", "wb") as fw:
    pickle.dump(data_total, fw, protocol = pickle.HIGHEST_PROTOCOL)

with open(r"H:\FYP\final\ShuffledLabel.pickle", "wb") as fw:
    pickle.dump(label_total, fw)

print("PROCESS DONE!")

### SPLITTING THE TRAINING AND TESTING DATASET IN 8:2 RATIO
with open(r"H:\FYP\final\ShuffledData.pickle", "rb") as fr:
    data_total = pickle.load(fr)

with open(r"H:\FYP\final\ShuffledLabel.pickle", "rb") as fr:
    label_total = pickle.load(fr)

### SPLITTING THE TRAINING AND TESTING DATASET IN 8:2 RATIO
training_set = int(len(data_total)*0.8)
test_set = int(len(data_total)*0.2)

data_training = data_total[0:training_set]
data_test = data_total[training_set:]
label_training = label_total[0:training_set]
label_test = label_total[training_set:]

```

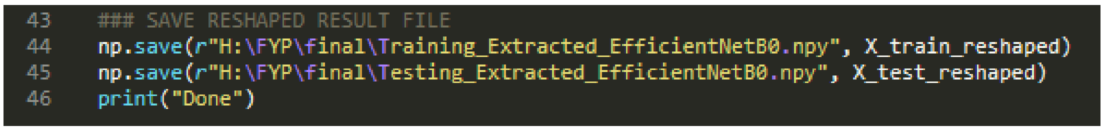


```
data_training[0].shape, label_training[0].shape
data_training[0][0, :, :, 0]

### SAVE TRAINING SET AND TEST SET AS PICKLE FILE
with open(r"H:\FYP\final\splitted dataset\TrainData.pickle", "wb") as fw:
    pickle.dump(data_training, fw, protocol = pickle.HIGHEST_PROTOCOL)

with open(r"H:\FYP\final\splitted dataset\TrainLabel.pickle", "wb") as fw:
    pickle.dump(label_training, fw)

with open(r"H:\FYP\final\splitted dataset\TestData.pickle", "wb") as fw:
    pickle.dump(data_test, fw, protocol = pickle.HIGHEST_PROTOCOL)

with open(r"H:\FYP\final\splitted dataset\TestLabel.pickle", "wb") as fw:
    pickle.dump(label_test, fw)

### TRANSFORMING ALL TRAINING DATASETS AS NUMPY ARRAYS
data_training_ar = np.array(data_training, dtype=np.float16)
np.save(r"H:\FYP\final\splitted dataset\TrainingData.npy", data_training_ar)

label_training_ar = np.array(label_training)
np.save(r"H:\FYP\final\splitted dataset\TrainingLabel.npy", label_training_ar)

data_training_ar.shape, label_training_ar.shape

### TRANSFORMING ALL TESTING DATASETS AS NUMPY ARRAYS
data_testing_ar = np.array(data_test, dtype=np.float16)
np.save(r"H:\FYP\final\splitted dataset\TestingData.npy", data_testing_ar)

label_testing_ar = np.array(label_test)
np.save(r"H:\FYP\final\splitted dataset\TestingLabel.npy", label_testing_ar)

data_testing_ar.shape, label_testing_ar.shape
```

- Feature Extraction (using EfficientNet B0)

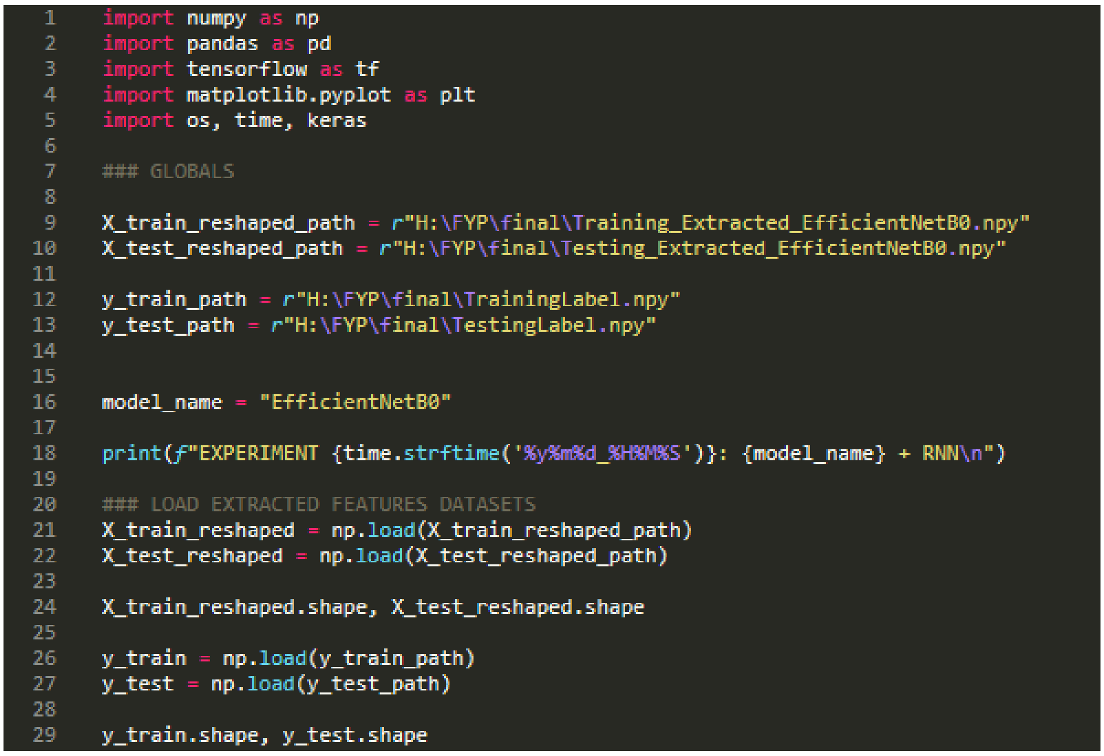


```
import numpy as np
import os, random, shufflenet
import tensorflow as tf

### LOAD DATASET ARRAYS
data_training_ar = np.load(r"H:\FYP\final\splitted dataset\TrainingData.npy")
data_test_ar = np.load(r"H:\FYP\final\splitted dataset\TestingData.npy")

data_training_ar.shape, data_test_ar.shape

### RESHAPING THE DATA
data_training_ar = data_training_ar.reshape(data_training_ar.shape[0]*30,160,160,3)
data_test_ar = data_test_ar.reshape(data_test_ar.shape[0]*30,160,160,3)

data_training_ar.shape, data_test_ar.shape

base_model=tf.keras.applications.EfficientNetB0(input_shape=(160,160,3), include_top=False,
                                                weights=None, input_tensor=tf.keras.layers.Input(shape=(160,160,3)))
base_model.summary()

### INSERT DATASET TO MOBILENETV2 BASE MODEL
np.random.seed(42)
X_train = base_model.predict(data_training_ar)
print(X_train.shape)
X_test = base_model.predict(data_test_ar)
print(X_test.shape)

### RESHAPE PREDICT RESULT TO INSERT LSTM
try:
    X_train_reshaped = X_train.reshape(int(X_train.shape[0]/30), 30,
                                int(X_train.shape[1])*int(X_train.shape[2])*int(X_train.shape[3]))
except:
    X_train_reshaped = X_train.reshape(int(X_train.shape[0]/30), 30, int(X_train.shape[1]))

try:
    X_test_reshaped = X_test.reshape(int(X_test.shape[0]/30), 30,
                                int(X_train.shape[1])*int(X_train.shape[2])*int(X_train.shape[3]))
except:
    X_test_reshaped = X_test.reshape(int(X_test.shape[0]/30), 30, int(X_train.shape[1]))

X_train_reshaped.shape, X_test_reshaped.shape
```

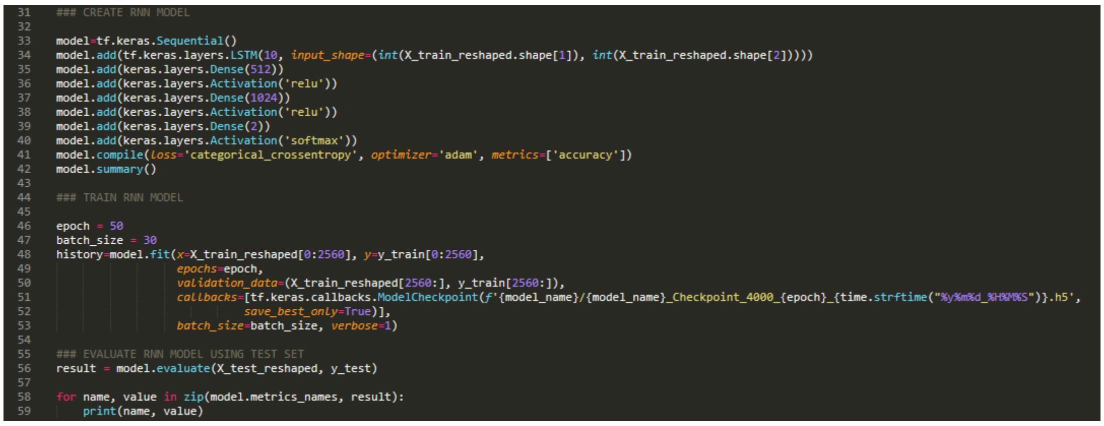


```
### SAVE RESHAPED RESULT FILE
np.save(r"H:\FYP\final\Training_Extracted_EfficientNetB0.npy", X_train_reshaped)
np.save(r"H:\FYP\final\Testing_Extracted_EfficientNetB0.npy", X_test_reshaped)
print("Done")
```

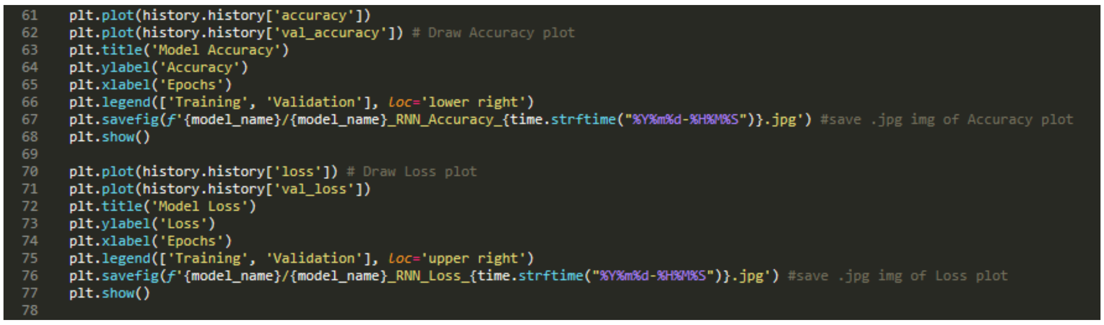
- Model Training



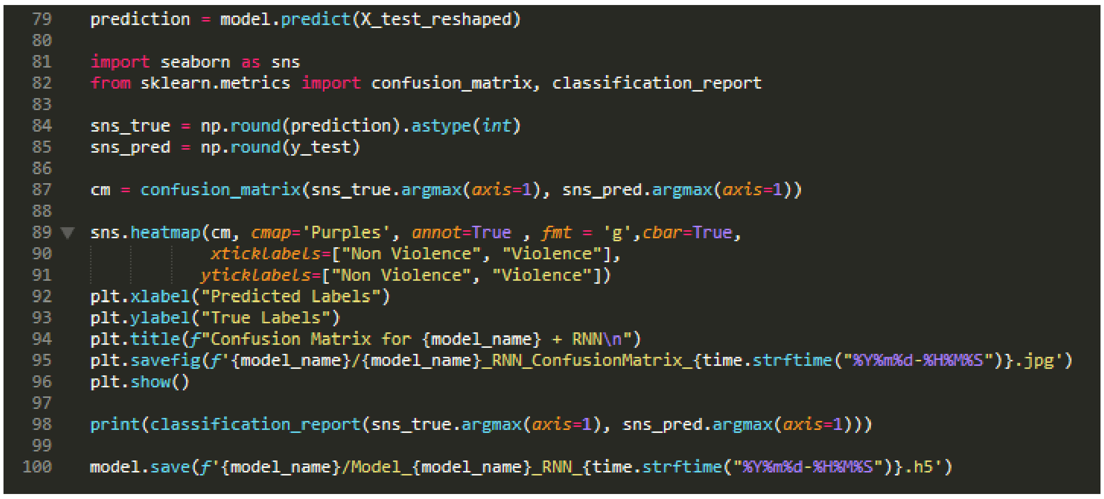
```python
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import os, time, keras

### GLOBALS

X_train_reshaped_path = r"H:\FYP\final\Training_Extracted_EfficientNetB0.npy"
X_test_reshaped_path = r"H:\FYP\final\Testing_Extracted_EfficientNetB0.npy"

y_train_path = r"H:\FYP\final\TrainingLabel.npy"
y_test_path = r"H:\FYP\final\TestingLabel.npy"


model_name = "EfficientNetB0"

print(f"EXPERIMENT {time.strftime('%y%m%d_%H%M%S')}: {model_name} + RNN\n")

### LOAD EXTRACTED FEATURES DATASETS
X_train_reshaped = np.load(X_train_reshaped_path)
X_test_reshaped = np.load(X_test_reshaped_path)

X_train_reshaped.shape, X_test_reshaped.shape

y_train = np.load(y_train_path)
y_test = np.load(y_test_path)

y_train.shape, y_test.shape
```



```python
### CREATE RNN MODEL

model=tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(10, input_shape=(int(X_train_reshaped.shape[1]), int(X_train_reshaped.shape[2]))))
model.add(keras.layers.Dense(512))
model.add(keras.layers.Activation('relu'))
model.add(keras.layers.Dense(1024))
model.add(keras.layers.Activation('relu'))
model.add(keras.layers.Dense(2))
model.add(keras.layers.Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()

### TRAIN RNN MODEL

epoch = 50
batch_size = 30
history=model.fit(x=X_train_reshaped[0:2560], y=y_train[0:2560],
                  epochs=epoch,
                  validation_data=(X_train_reshaped[2560:], y_train[2560:]),
                  callbacks=[tf.keras.callbacks.ModelCheckpoint(f'{model_name}/{model_name}_Checkpoint_4000_{epoch}_{time.strftime("%y%m%d_%H%M%S")}.h5',
                            save_best_only=True)],
                  batch_size=batch_size, verbose=1)

### EVALUATE RNN MODEL USING TEST SET
result = model.evaluate(X_test_reshaped, y_test)

for name, value in zip(model.metrics_names, result):
    print(name, value)
```

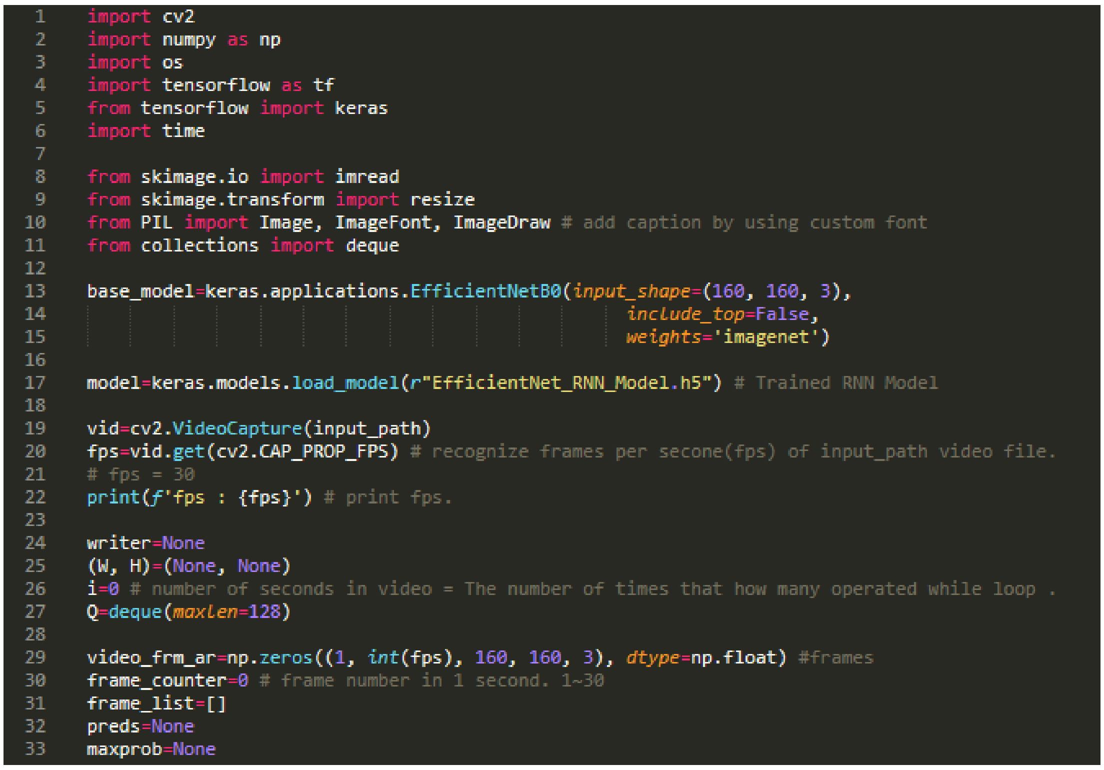


```python
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy']) # Draw Accuracy plot
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Training', 'Validation'], loc='lower right')
plt.savefig(f'{model_name}/{model_name}_RNN_Accuracy_{time.strftime("%Y%m%d-%H%M%S")}.jpg') #save .jpg img of Accuracy plot
plt.show()

plt.plot(history.history['loss']) # Draw Loss plot
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Training', 'Validation'], loc='upper right')
plt.savefig(f'{model_name}/{model_name}_RNN_Loss_{time.strftime("%Y%m%d-%H%M%S")}.jpg') #save .jpg img of Loss plot
plt.show()
```

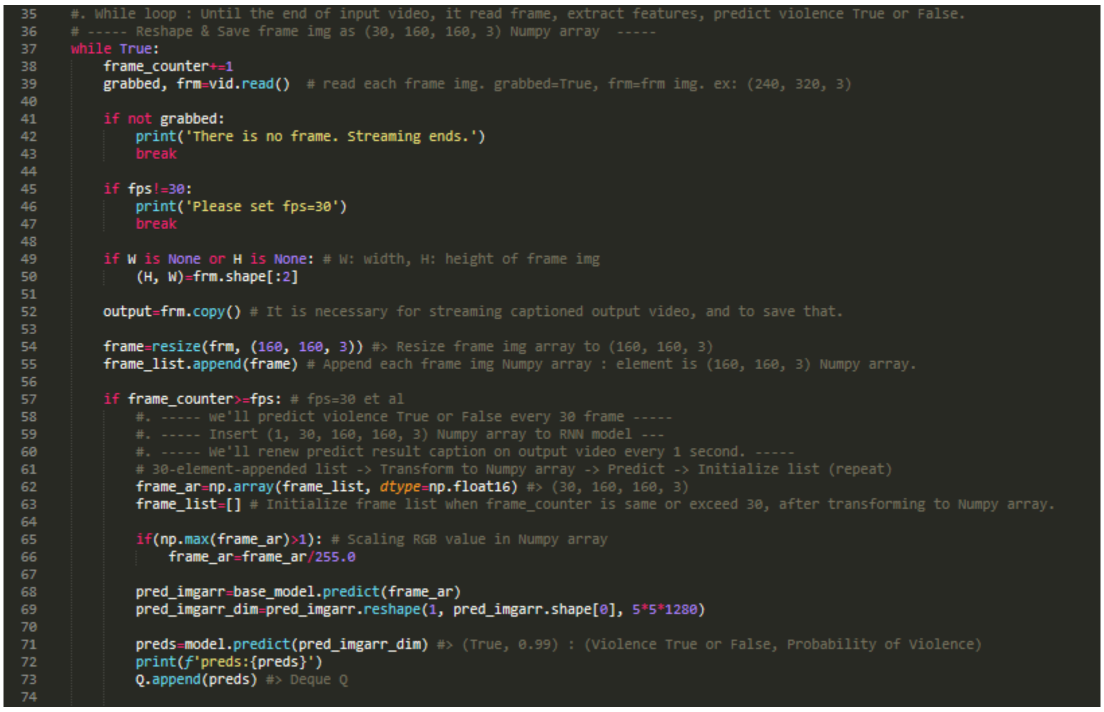


```python
prediction = model.predict(X_test_reshaped)

import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report

sns_true = np.round(prediction).astype(int)
sns_pred = np.round(y_test)

cm = confusion_matrix(sns_true.argmax(axis=1), sns_pred.argmax(axis=1))

sns.heatmap(cm, cmap='Purples', annot=True , fmt = 'g',cbar=True,
            xticklabels=["Non Violence", "Violence"],
           yticklabels=["Non Violence", "Violence"])
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title(f"Confusion Matrix for {model_name} + RNN\n")
plt.savefig(f'{model_name}/{model_name}_RNN_ConfusionMatrix_{time.strftime("%Y%m%d-%H%M%S")}.jpg')
plt.show()

print(classification_report(sns_true.argmax(axis=1), sns_pred.argmax(axis=1)))

model.save(f'{model_name}/Model_{model_name}_RNN_{time.strftime("%Y%m%d-%H%M%S")}.h5')
```

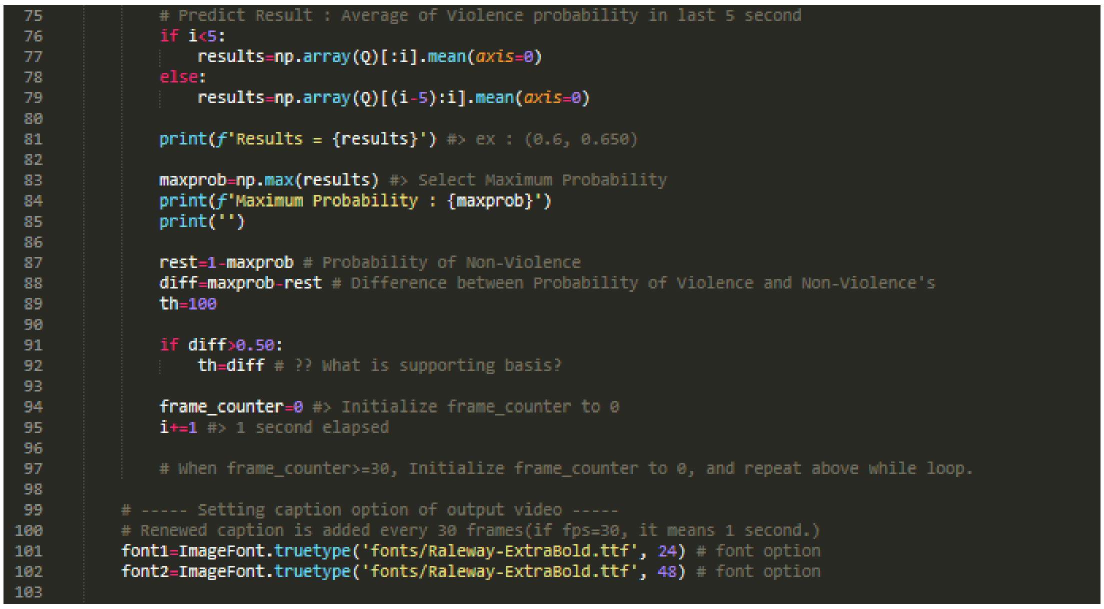
- Real Time Inferencing



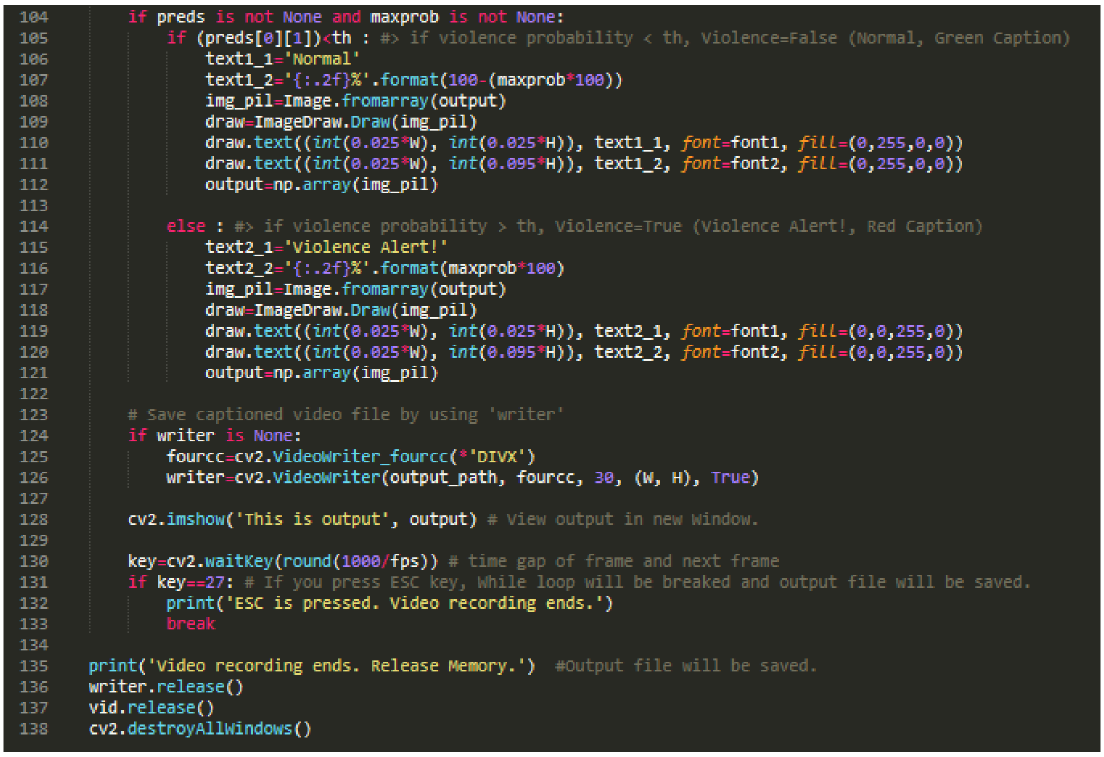
```python
import cv2
import numpy as np
import os
import tensorflow as tf
from tensorflow import keras
import time

from skimage.io import imread
from skimage.transform import resize
from PIL import Image, ImageFont, ImageDraw # add caption by using custom font
from collections import deque

base_model=keras.applications.EfficientNetB0(input_shape=(160, 160, 3),
                                              include_top=False,
                                              weights='imagenet')

model=keras.models.load_model(r"EfficientNet_RNN_Model.h5") # Trained RNN Model

vid=cv2.VideoCapture(input_path)
fps=vid.get(cv2.CAP_PROP_FPS) # recognize frames per secone(fps) of input_path video file.
# fps = 30
print(f'fps : {fps}') # print fps.

writer=None
(W, H)=(None, None)
i=0 # number of seconds in video = The number of times that how many operated while loop .
Q=deque(maxlen=128)

video_frm_ar=np.zeros((1, int(fps), 160, 160, 3), dtype=np.float) #frames
frame_counter=0 # frame number in 1 second. 1~30
frame_list=[]
preds=None
maxprob=None
```



```python
#. While loop : Until the end of input video, it read frame, extract features, predict violence True or False.
# ----- Reshape & Save frame img as (30, 160, 160, 3) Numpy array  -----
while True:
    frame_counter+=1
    grabbed, frm=vid.read()  # read each frame img. grabbed=True, frm=frm img. ex: (240, 320, 3)

    if not grabbed:
        print('There is no frame. Streaming ends.')
        break

    if fps!=30:
        print('Please set fps=30')
        break

    if W is None or H is None: # W: width, H: height of frame img
        (H, W)=frm.shape[:2]

    output=frm.copy() # It is necessary for streaming captioned output video, and to save that.

    frame=resize(frm, (160, 160, 3)) #> Resize frame img array to (160, 160, 3)
    frame_list.append(frame) # Append each frame img Numpy array : element is (160, 160, 3) Numpy array.

    if frame_counter>=fps: # fps=30 et al
        #. ----- we'll predict violence True or False every 30 frame -----
        #. ----- Insert (1, 30, 160, 160, 3) Numpy array to RNN model ---
        #. ----- We'll renew predict result caption on output video every 1 second. -----
        # 30-element-appended list -> Transform to Numpy array -> Predict -> Initialize list (repeat)
        frame_ar=np.array(frame_list, dtype=np.float16) #> (30, 160, 160, 3)
        frame_list=[] # Initialize frame list when frame_counter is same or exceed 30, after transforming to Numpy array.

        if(np.max(frame_ar)>1): # Scaling RGB value in Numpy array
            frame_ar=frame_ar/255.0

        pred_imgarr=base_model.predict(frame_ar)
        pred_imgarr_dim=pred_imgarr.reshape(1, pred_imgarr.shape[0], 5*5*1280)

        preds=model.predict(pred_imgarr_dim) #> (True, 0.99) : (Violence True or False, Probability of Violence)
        print(f'preds:{preds}')
        Q.append(preds) #> Deque Q

```



```
        # Predict Result : Average of Violence probability in last 5 second
        if i<5:
            results=np.array(Q)[:i].mean(axis=0)
        else:
            results=np.array(Q)[(i-5):i].mean(axis=0)

        print(f'Results = {results}') #> ex : (0.6, 0.650)

        maxprob=np.max(results) #> Select Maximum Probability
        print(f'Maximum Probability : {maxprob}')
        print('')

        rest=1-maxprob # Probability of Non-Violence
        diff=maxprob-rest # Difference between Probability of Violence and Non-Violence's
        th=100

        if diff>0.50:
            th=diff # ?? What is supporting basis?

        frame_counter=0 #> Initialize frame_counter to 0
        i+=1 #> 1 second elapsed

        # When frame_counter>=30, Initialize frame_counter to 0, and repeat above while loop.

    # ----- Setting caption option of output video -----
    # Renewed caption is added every 30 frames(if fps=30, it means 1 second.)
    font1=ImageFont.truetype('fonts/Raleway-ExtraBold.ttf', 24) # font option
    font2=ImageFont.truetype('fonts/Raleway-ExtraBold.ttf', 48) # font option

```



```
    if preds is not None and maxprob is not None:
        if (preds[0][1])<th : #> if violence probability < th, Violence=False (Normal, Green Caption)
            text1_1='Normal'
            text1_2='{:.2f}%'.format(100-(maxprob*100))
            img_pil=Image.fromarray(output)
            draw=ImageDraw.Draw(img_pil)
            draw.text((int(0.025*W), int(0.025*H)), text1_1, font=font1, fill=(0,255,0,0))
            draw.text((int(0.025*W), int(0.095*H)), text1_2, font=font2, fill=(0,255,0,0))
            output=np.array(img_pil)

        else : #> if violence probability > th, Violence=True (Violence Alert!, Red Caption)
            text2_1='Violence Alert!'
            text2_2='{:.2f}%'.format(maxprob*100)
            img_pil=Image.fromarray(output)
            draw=ImageDraw.Draw(img_pil)
            draw.text((int(0.025*W), int(0.025*H)), text2_1, font=font1, fill=(0,0,255,0))
            draw.text((int(0.025*W), int(0.095*H)), text2_2, font=font2, fill=(0,0,255,0))
            output=np.array(img_pil)

    # Save captioned video file by using 'writer'
    if writer is None:
        fourcc=cv2.VideoWriter_fourcc(*'DIVX')
        writer=cv2.VideoWriter(output_path, fourcc, 30, (W, H), True)

    cv2.imshow('This is output', output) # View output in new Window.

    key=cv2.waitKey(round(1000/fps)) # time gap of frame and next frame
    if key==27: # If you press ESC key, While loop will be breaked and output file will be saved.
        print('ESC is pressed. Video recording ends.')
        break

print('Video recording ends. Release Memory.')  #Output file will be saved.
writer.release()
vid.release()
cv2.destroyAllWindows()
```