CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY

**GRADUATION THESIS**
**BACHELOR OF ENGINEERING IN INFORMATION TECHNOLOGY**
**(HIGH-QUALITY PROGRAM)**

# BUILDING A MODEL FOR DETECTING VIOLENT BEHAVIOR IN CHILDREN

**Student: Phan Ba Dai Phuc**
**Student ID: B1910688**
**Class: 2019-2023 (Course 45)**
**Advisor: Dr. Pham Thi Ngoc Diem**

**Can Tho, 12/2023**

CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY

GRADUATION THESIS
BACHELOR OF ENGINEERING IN
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)

# BUILDING A MODEL FOR DETECTING VIOLENT BEHAVIOR IN CHILDREN

Student: Phan Bá Đại Phúc
Student ID: B1910688
Class: 2019-2023 (Course 45)
Advisor: Dr. Pham Thi Ngoc Diem

Can Tho, 12/2023

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

**KHOA CÔNG NGHỆ THÔNG TIN**

# XÁC NHẬN CHỈNH SỬA LUẬN VĂN THEO YÊU CẦU CỦA HỘI ĐỒNG

Tên luận văn: Xây dựng mô hình phát hiện hành vi bạo hành trẻ em

Họ tên sinh viên: Phan Bá Đại Phúc MASV: B1910688

Mã lớp: DI19V7F1

Đã báo cáo tại hội đồng ngành: Công nghệ thông tin (Chất lượng cao)

Ngày báo cáo: 04/12/2023

Hội đồng báo cáo gồm:

| | |
|---|---|
| 1. T.S. Trần Công Án | Chủ tịch hội đồng |
| 2. T.S. Bùi Võ Quốc Bảo | Thành viên |
| 3. T.S. Phạm Thị Ngọc Diễm | Thư ký |

Luận văn đã được chỉnh sửa theo góp ý của Hội đồng.

*Cần Thơ, ngày 04 tháng 12 năm 2023*

**Giáo viên hướng dẫn**

*(Ký và ghi họ tên)*

COLLEGE OF INFORMATION COMMUNICATION TECHNOLOGY

**INFORMATION TECHNOLOGY**

# CONFIRMATION OF THESIS MODIFICATION BASED ON THE REQUESTS OF THE COMMITTEE

Thesis name: Building A Model For Detecting Violent Behavior In Children

Student: Phan Ba Dai Phuc Student ID: B1910688

Class ID: DI19V7F1

Reported at the committee: Information technology (High-quality program)

Report date: 04/12/2023

The committee includes:

| | |
|---|---|
| 1. Dr. Tran Cong An | President |
| 2. Dr. Bui Vo Quoc Bao | Member |
| 3. Dr. Pham Thi Ngoc Diem | Secretary |

This thesis has been modified according to the opinion of the committee.

*Can Tho, December 04th, 2023*

**Advisor**

*(Signature and full name)*

## COMMENTS OF ADVISOR

------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------

Can Tho, …………………… 2023

**Advisor**

*(Signature and full name)*

## ACKNOWLEDGMENT

*I cannot express enough thanks to my instructors/teachers from the College of Information and Communication Technology for their continued supports and encouragement. My sincere thanks especially go to Dr. Phạm Thị Ngọc Diễm for her advice and guidance throughout the development of this project.*

*I could not have achieved the completion of this project without the support of my classmates. The support and advice from them has always been the greatest motivation for me throughout the process. I greatly appreciate and duly acknowledge their encouragement during difficult times.*

*Last but not least, I would like to thank my parents, sisters, niece and cousin who helped me a lot in gathering various information, helping in my data collection and guiding me. Despite their busy schedules, they were always by my side when it comes to completing the project.*

Can Tho, December 2023

Student

Phan Bá Đại Phúc

## ABSTRACT

Child abuse is a grave and pervasive social problem with profound consequences for both individual victims and society as a whole. However, there are not many models or application that available in order to detect children violence. Proper detection is important not only in recognizing children abuse but also apply appropriate penalties for those who perform violence on children. Efforts to enhance child abuse detection are crucial in addressing this sensitive issue and ensuring the safety and well-being of vulnerable children. As technology and interdisciplinary collaboration continue to evolve, I spend my endeavor to develop a machine learning model which help detecting children violence. In my project, I use machine learning to provide detailed analysis and detect whether there is any children violence is performed in videos.

In addition, Deep Learning and Computer Vision are being intensively researched and improved every day. In particular, Google's development of MediaPipe, an open-source framework for building world-class machine learning solutions that provide basic machine learning models for common tasks such as hand tracking, posture recognition, ... Another famous framework that I used in this thesis is YOLOv8 which is well-known for its efficiency in object detection, image segmentation, … This project, "Children abuse detection in videos based on machine learning", is based on the detection of postures by MediaPipe and YOLOv8 which are used to analyze, detect and classify actions of violence. The final experimental results show that the algorithm proposed in this work can effectively identify which actions are performed in videos and in realtime.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# INTRODUCTION

## 1. PROBLEM DESCRIPTION

With the advances in artificial intelligence and computing power, computer vision technology has made a great leap towards integration into our daily lives. Computer vision is a branch of computer science that deals with the development of digital systems that can process, analyze, and interpret visual data (images or videos) in the same way that humans do. The concept of computer vision is based on teaching computers to process and understand an image at the pixel level. Technically, machines attempt to retrieve visual information, process it, and interpret the results through special software algorithms.

Children is the future of our society as they are the one who inherit our achievements, knowledge and apply them to contribute to a better world. Therefore, protecting children from violence, harmful activities is our top priority in any circumstances. However, nowadays, number of children who suffer from violence, molested, harmful activities are tremendously at high rate. Hence, approaches in order to minimize this demerits should be applied at wide range. Children violence detection is of paramount importance as it is safeguarding of vulnerable individuals. These young individuals are often unable to protect themselves from violence, making it imperative for society to intervene on their behalf. Furthermore, detecting children violence is the first step in preventing further harm to the children. This harm can be physical, emotional, or psychological. Early detection and intervention can prevent long-term trauma and suffering. In many countries

This thesis is about building machine learning models for different violent actions using the MediaPipe framework and the alternative approach is YOLOv8. For each video input, models will detect whether there is any violence action is performed in this video. If we are desire to use realtime detection, MediaPipe framework will take the input from webcam and detect realtime.

## 2. PURPOSE OF THE STUDY

The goal of this study is to develop 4 machine learning models for 4 of the most common violent actions in children, where each model can detect any violent action is performed on children.

## 3. LIMITATION AND SCOPE

This study researches deep learning knowledge such as computer vision, neural network and MediaPipe framework. The study also uses Python programming language, OpenCV library for image processing; Sci-kit learn library and Keras library for building machine learning model; IPython library for explicitly displaying objects like images,

## 4. GENERAL APPROACH

- ❖ Research on which popular violence action which commonly perform in children
- ❖ Research on which technology to choose that is suitable to solve the problem.
- ❖ Collect and process data of the chosen actions.
- ❖ Train and evaluate model in MediaPipe and YOLO.

## 5. CRITERIA FOR STUDY SUCCESS

- ❖ Successfully build 5 models for 4 actions which can detect violence action (if exists) or show that it is normal.
- ❖ Earn knowledge on deep learning and computer vision topics. Apply MediaPipe framework and YOLO to real life issues.

## 6. OUTLINE OF THE STUDY

- ❖ Introduction: Introduce the problem and the general concepts of the topic, such as expectation, limit, content, etc.
- ❖ Chapter 1: Related theories and tools. Present related theoretical content about computer vision, the MediaPipe framework, YOLOv8 model, and other technologies used in the study.
- ❖ Chapter 2: Methodology. Present general methods for detecting violence in the selected actions.
- ❖ Chapter 3: Results. Present evaluation metrics, evaluation results for all trained models, and create test scenarios, test results for models.
- ❖ Chapter 4: Conclusion

## 7. RELATED WORKS

1. On September 21, 2022, a project called "*Violence Detection Between Children and Caregivers Using Computer Vision*"[1] was released in order to detect violence in children. This research focused on developing an automated system to detect violent interactions between children and caregivers using computer vision techniques. The researchers trained a machine learning model on a dataset of video clips sourced from YouTube. This project apply RestNet50 as the convolutional layers to extract features and LSTM and CNN for the improvement of data

persistence. The model achieved an overall accuracy of 85% in identifying violent interactions.

2. Another related research is about detecting violence in adult and children interaction called "*Multimodal Violence Detection in Child-Adult Interactions*" by Cao et al. (2019) [2]. This research investigates the combination of audio, visual, and contextual information to enhance the detection of violence in child-adult interactions. The researchers collect multimodal data from child-adult interactions and employ deep learning techniques to extract relevant features from each modality. They then fuse the extracted features and train a machine learning model for violence detection. The multimodal approach achieves an accuracy of 89.4%.

3. "*Audio-Based Violence Detection in Everyday Child-Adult Interactions*" [3] is a research that explored the use of audio features to detect violence in everyday interactions between children and adults. The researchers collected audio recordings of interactions and extracted features such as pitch, energy, and speech rate. They employed machine learning algorithms to classify interactions as either violent or non-violent. The best-performing classifier achieved an accuracy of 78.6%.

4. "*Leveraging Social Signals for Violence Detection in Children*" [4] concentrated on utilizing social signals, such as facial expressions, body language, and gaze, to detect violence in children. The researchers collected video recordings of children's interactions and extracted social signal features using computer vision techniques. They then employed machine learning algorithms to classify interactions as either violent or non-violent. The best-performing classifier achieved an accuracy of 82.5%.

5. "*Campus Violence Detection Based on Artificial Intelligent Interpretation of Surveillance Video Sequences*" [5] use image features and acoustic features for campus violence detection. Campus violence data are gathered by role-playing, and 4096-dimension feature vectors are extracted from every 16 frames of video images. The C3D (Convolutional 3D) neural network is used for feature extraction and classification, and an average recognition accuracy of 92.00% is achieved. Mel-frequency cepstral coefficients (MFCCs) are extracted as acoustic features, and three speech emotion databases are involved. The C3D neural network is used for classification, and the average recognition accuracies are 88.33%, 95.00%, and 91.67%, respectively.

**CONTENTS**

# CHAPTER 1. RELATED THEORIES AND TOOLS

## 1.1. COMPUTER VISION

Computer vision is the field of computer science that focuses on creating digital systems that can process, analyze, and make sense of visual data (images or videos) in the same way that humans do. The concept of computer vision is based on teaching computers to process an image at a pixel level and understand it. Technically, machines attempt to retrieve visual information, handle it, and interpret results through special software algorithms.

There is a lot of research being done in the computer vision field, but it’s not just research. Real-world applications demonstrate how important computer vision is to endeavors in business, entertainment, transportation, healthcare and everyday life. A key driver for the growth of these applications is the flood of visual information flowing from smartphones, security systems, traffic cameras and other visually instrumented devices.

Google Translate lets users point a smartphone camera at a sign in another language. IBM is applying computer vision technology with partners like Verizon to bring intelligent AI to the edge, and to help automotive manufacturers identify quality defects before a vehicle leaves the factory.

The development of self-driving vehicles relies on computer vision to make sense of the visual input from a car’s cameras and other sensors. It’s essential to identify other cars, traffic signs, lane markers, pedestrians, bicycles and all of the other visual information encountered on the road.

## 1.2. DEEP LEARNING

Deep learning is a sub-branch of machine learning field, concentrating primarily on solving issues that related to neural network, it can also be comprehended as the brain of computer for smart system as it can learn from diverse datasets. The optimal model will be selected to apply into actual project with different purposes. Every model will have specific aims and particular application. For instance, some of the actual projects that deep learning model include automatic driving system, image recognition, converting audio to text, and medical diagnostic.

In the field of Computer Vision, there are various problems such as classification tasks, object localization tasks, object detection tasks, and image segmentation tasks.

**Classification:** This is the problem of predicting label of object when the input image include object and the output is the label of the object.

**Object localization:** this is the problem of identifying position of object, the object detected will have rounded box around it.
**Object detection**: is a combination of 2 classification and localization, but there is a major difference which is object detection detect not only single object in input image but multiple objects.
**Image Segmentation**: used to segment objects into different parts.

The "Children Violence Detection" project concentrate on studying the detection of people who have been detected of performing violence action on children.

### 1.2.1 Convolutional Neural Network

Nowadays, there are a lot of smart system invented in order to serve our daily purposes due to its efficiency, accelerated computing rate and precision of CNN. It was first published in a scientific conference in 1998 by Yan Lecun.
In the technological aspect, CNN comprises 3 primary components: Convolutional Layer, Pooling Layer and Fully Connected Layer.

#### Convolutional Layer

This is the first layer and also the most essential in the CNN model as its purpose is to extract features out of the input image.
Convolutional Layer takes the input image, divides that image into 3 matrices (Red, Green, Blue) and apply the convolution operation which is the dot multiplication with the kernel with specific stride in order to get the feature map.

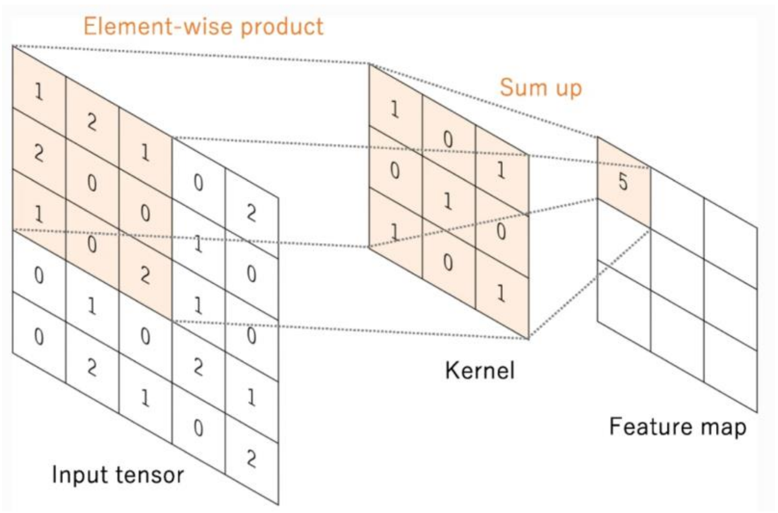


***Figure 1.*** *Example of convolution a 5x5 matrix with a 3x3 filter [6]*

According to **Figure 1**, it shows the input matrix, kernel and the feature map. At first, the a matrix with the size of kernel is selected in the input matrix and then being dot multiplied and written in the first element of the feature map, the next selected matrix is chosen on the right side of the input matrix depending on the stride initialized. The process is accomplished if and only if the feature map is fully filled for the next step.

**Stride:** is the step size of the kernel after every time convoluted on every point. For instance, Stride is 2, then the kernel will move 2 cells on the right side of the input matrix.

**Padding:** padding is referred to the border of the matrix as sometimes the input matrix does not fit the kernel and stride. Therefore, by adding padding (usually 0 or 1), will make the calculation far easier. An example of stride and padding is demonstrated in the **Figure 2**

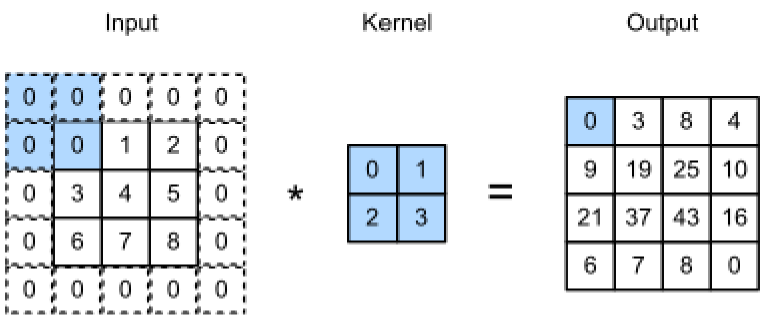


***Figure 2.*** *Example with stride = 2, padding = 0 [7]*

**Relu function**: this is a linear function which used mostly in training model. It purpose is to eliminate the negative value in matrices. The general formula is:

$$f(x) = max(0, x)$$

**Pooling Layer**

Pooling layer is utilized after the convolutional layer is performed by reducing the input for the next process. However, this reduction will certainly maintain the key feature of the image.

There are 3 common types of pooling methods including Max Pooling, Average Pooling and Sum Pooling. Max Pooling is the most well-known one among three.

**Max Pooling**: return the maximum value in the selected matrix. **Figure 3** will demonstrate the Max Pooling process.

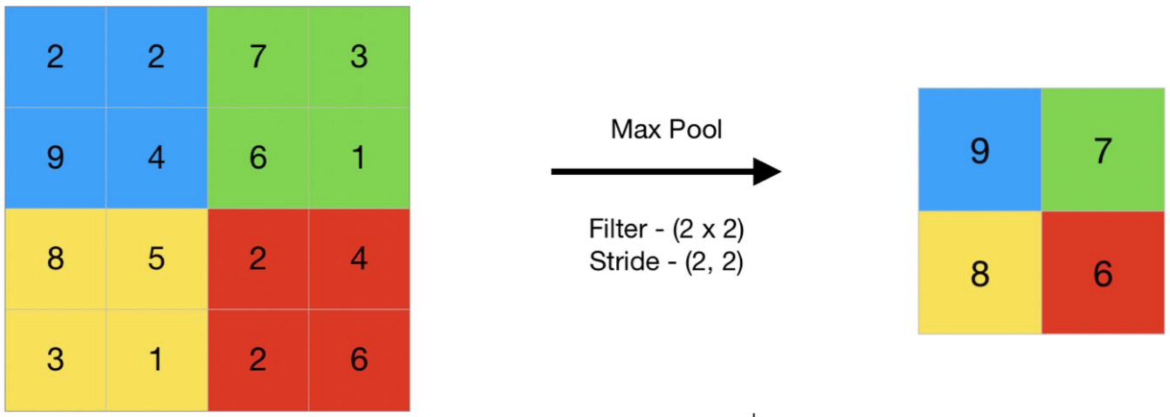


***Figure 3.*** *Example of pooling layer using max pooling [8]*

**Fully Connected Layer:**

Fully Connected Layer creates a model and classifies the output and it is similar to the **neural network**. After the Pooling Layer process, the matrix will now be flattened by transfer to a vertex vector. The vector will now be connected to the neurons through the weights (trained with the provided dataset). After passing through the neurons, an activation function such as **sigmoid, or tanh** will be applied to the final neurons layer and it will return a vector which includes the probability of every class. The process will be illustrated in **Figure 4**

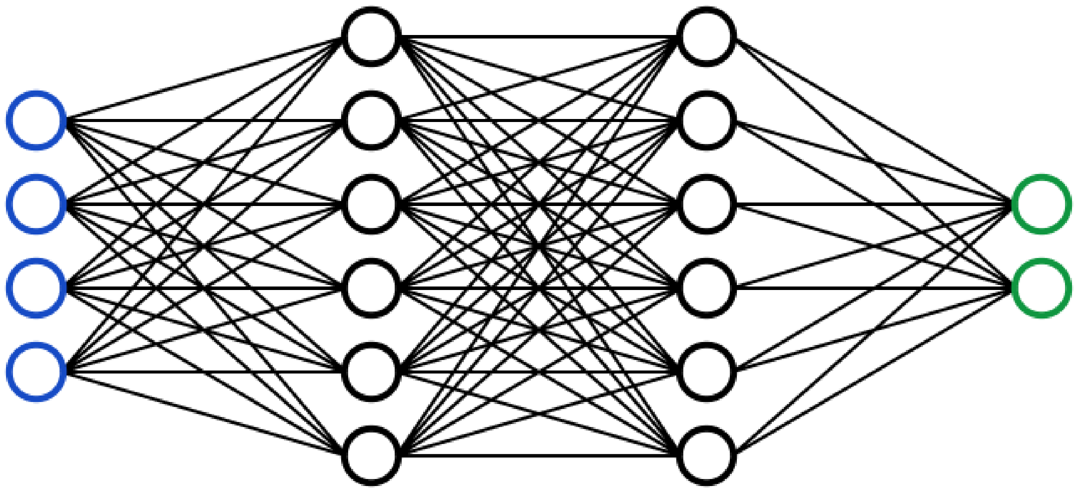


***Figure 4.*** *Fully connected and flattened data [9]*

**Training process**

The training process is similar to the predicting process. However, there are some major disparity as the weights in training process is randomly initialized or pre-trained in some certain models. Next, the result will be compared to the ground truth (which is the actual result trainer provides), the **Loss function** will be calculated. After that, the weights and other parameters will be updated through the backpropagation process. In this process, the optimal weights will be calculated using **gradient descent** process.

### 1.2.2 YOLO model

Nowadays, YOLO has developed to the 8th edition (YOLOv8), every version offers high efficiency, different improvements. The accuracy rate of YOLO still needs to be improved, yet the processing speed is remarkably enhanced. YOLO is an algorithm solving the issue of positioning objects, classifying objects.
YOLO is a sub network of CNN. The convolutional layer of CNN is replaced by Darknet structure as the base layers whose purpose is to extract the feature of the input image and the output will be the feature map.

**Training process**: YOLO is a sub-network of CNN in which the convolutional layers are replaced with **Darknet** structure as the fundamental tool for feature extraction. The model performs the convolutional operations which apply the dot multiplication for the kernel and selected matrix. The kernel will move from left to right and vertically with the stride has been defined before. Down sample operations namely Max-Pooling, Average-Pooling, Sum-Pooling (mostly use Max-Pooling) are performed in order to reduce the input for the next operations. The output of this process is a feature map of 7x7x1024 in size and flattened in a vertical vector of 4096 in size and passed via the fully connected layers. After the 2 fully connected layers, the output tensor is a matrix of 7x7x30, this matrix is used to predict and calculate the probability of labels.

**Detecting process**: YOLO takes an image as an input and divides it into a grid of cells of SxS in size and applies the kernels. Next, each grid cell predicts the bounding box based on confident scores. Objects belong to which cell, it will be responsible for detecting that object. Every cell is programmed to predict 2 bounding boxes and probability with the condition of object existence. The initial dataset used for the pre-training process is PASCAL VOC with 20 classes vary in many fields (animals, vehicles, objects). Each bounding box is required to predict 5 values including confidence score, x, y, w, h where (x, y) are coordinates of the center of bounding box, and (w, h) are width and height of bounding box compared to image size (not cell). Confidence score represents the probability of an object present in that cell. If the bounding box does not contain an object, confidence, therefore, will be 0. Otherwise, it is calculated by the multiplication of probability of containing object and IoU scores (Intersection over Units). To sum up, every cell needs to predict B bounding boxes and probability of C classes. Therefore, the number of dimension in tensor is calculated by this formula:

$$S \times S \times (5B + C)$$

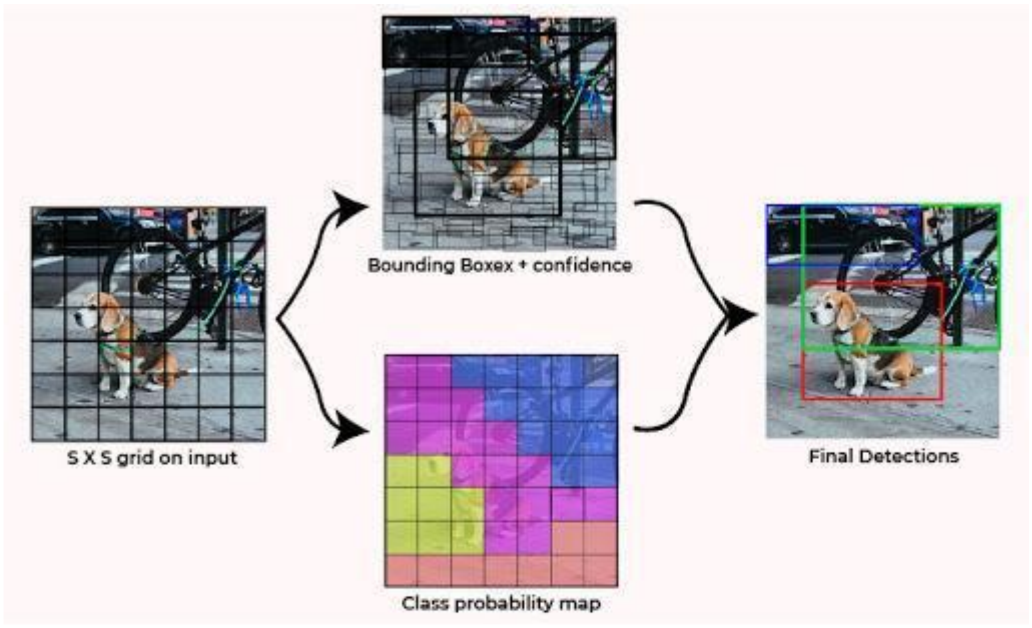


***Figure 5.*** *System divided input image to SxS grid [10]*

**1.2.3 YOLOv5 model**

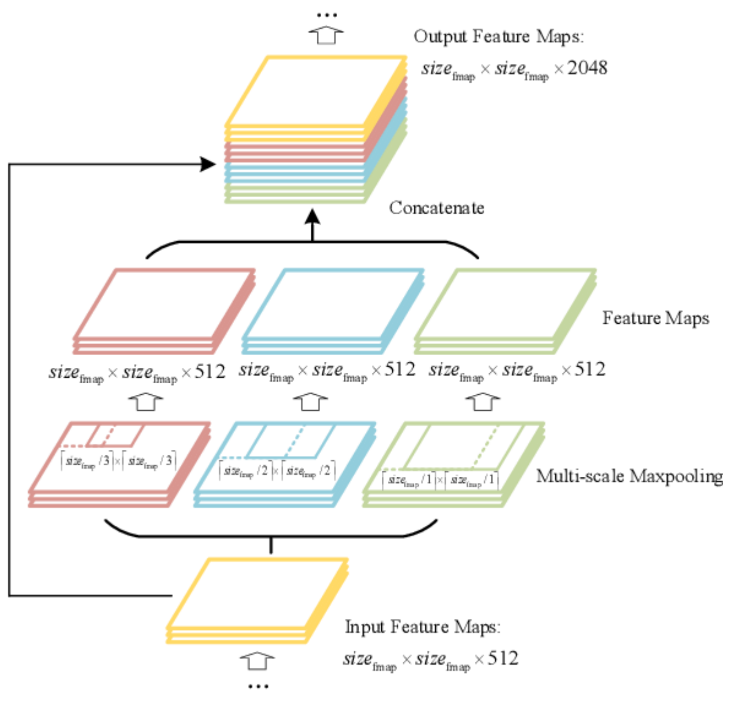
YOLOv5 (You Only Look Once v5) is a state-of-the-art object detection algorithm that utilizes a convolutional neural network (CNN) architecture for image classification and localization. Its architecture consists of three main components: the backbone, the neck, and the head.

**Backbone**

The backbone is the foundation of the YOLOv5 model, responsible for extracting high-level features from the input image. It utilizes the CSPDarknet53 architecture, a modified version of the Darknet53 network, known for its efficiency and accuracy. CSPDarknet53 incorporates several enhancements, including:

Cross Stage Partial connections (CSP): CSP connections reduce the number of parameters and computations without compromising accuracy.

Spatial Pyramid Pooling (SPP): SPP allows the network to learn features from different scales of the input image. This network is used in YOLOv3 (YOLOv3-SPP) and continuously utilized until the current version. However, with YOLOv3, the SPP network receives feature map input with different kernel sizes and input feature maps are divided into parts using Multi-scale Maxpooling to get the feature maps with the same size using Concatenate combined with feature maps received. YOLOv4 does not use that approach of extracting features. It applies convolutional Max-pooling to get the feature map and transfer to the output which is stacking illustrated in **Figure 6**.


Output Feature Maps:
$size_{fmap} \times size_{fmap} \times 2048$
Concatenate
Feature Maps
$size_{fmap} \times size_{fmap} \times 512$
Multi-scale Maxpooling
Input Feature Maps:
$size_{fmap} \times size_{fmap} \times 512$

(b)

***Figure 6.** YOLO-SPP Model [11]*

Path Aggregation Network (PAN): PAN enables the network to combine features from different levels of the backbone to improve detection accuracy. PAN is an upgraded of FPN model. PAN allows objects which have minimal size after every downsample are improved and minimize loss information due to top-down and bottom-up technique in which PAN is attached with bottom-up and bottom-down layers of FPN (N4) demonstrated in **Figure 7**.

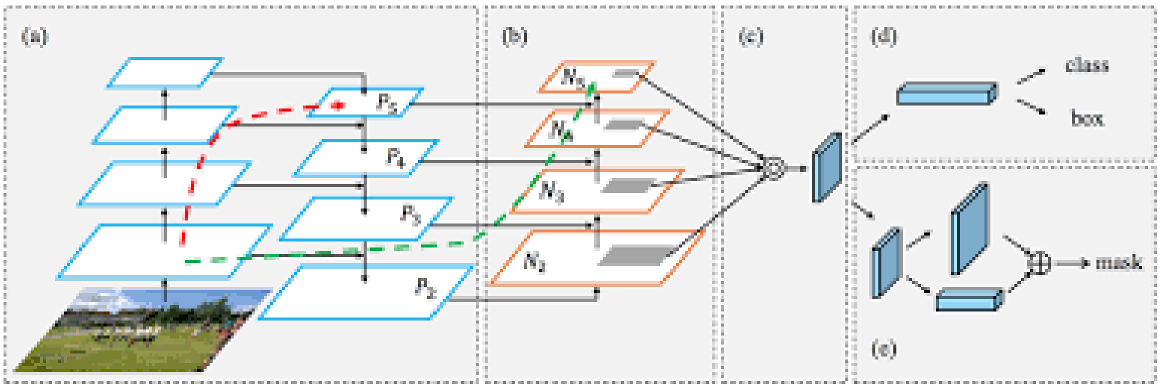


***Figure 7.*** *PAN model [12]*

In (b) process in **Figure 7**, those layers will receive input as feature maps of the former layer and passing through a matrix 3x3. Output is concatenated with a feature map of top-down pathway.

**Neck**

The neck connects the backbone to the head, responsible for fusing features from different layers of the backbone to provide a more comprehensive representation of the input image. YOLOv5 employs a modified version of the Path Aggregation Network (PAN) as its neck. The modified PAN utilizes the fused features to predict bounding boxes and object classes at different scales.

**Head**

The head is the final stage of the YOLOv5 architecture, responsible for predicting bounding boxes and object classes for each object detected in the input image. It consists of a series of convolutional and linear layers. The convolutional layers refine the features from the neck, while the linear layers predict the bounding boxes, object classes, and confidence scores for each detected object.

In previous versions of YOLO, such as YOLOv3, logistic regression was used to predict the objectness score for each bounding box. The objectness score represented the model's confidence in whether there was an object present in a particular grid cell of the input image. However, in YOLOv5, this approach was replaced with a more direct prediction of bounding box coordinates and object class probabilities.

YOLOv5 does not directly use logistic regression to predict the object score for each object. Instead, it utilizes a combination of convolutional and linear layers to predict the bounding box center coordinates, width, height, and object class probabilities. The object score, also known as the confidence score, is derived from the predicted object class probabilities.

The reason for this change is that logistic regression is not well-suited for predicting continuous values like bounding box coordinates and probabilities. Convolutional and linear layers, on the other hand, are specifically designed to handle such tasks. By using these layers directly, YOLOv5 can achieve more accurate and precise detection results.

**Data Augmentation Techniques**

YOLOv5 employs various data augmentation techniques to improve the model's ability to generalize and reduce overfitting. These techniques include:

Random Mosaic: Mosaic Data Augmentation is a simple augmentation technique in which 4 images are randomly stitched together and used as input images illustrated in **Figure 8**. Model will therefore be able to learn actual objects in disparate positions.

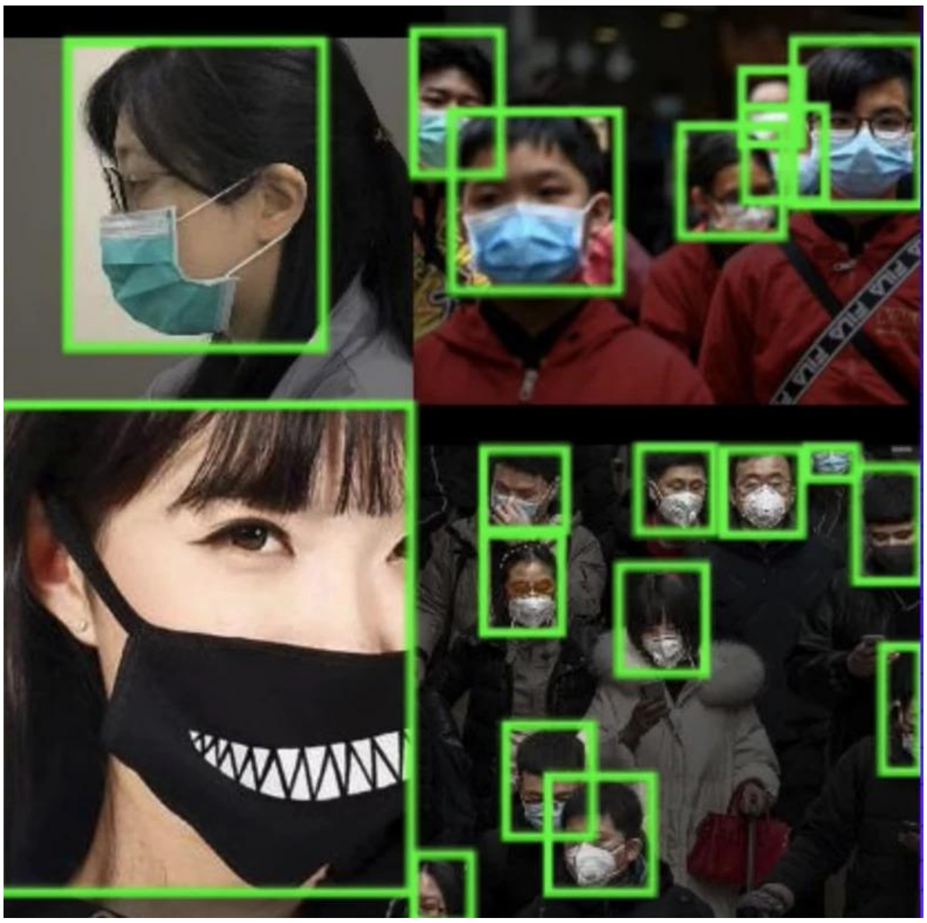


***Figure 8.*** *Data augmentation using Mosaic approach of mask detection [13]*

**MixUp**: MixUp combines two images and corresponding labels to create a new image and label, further enriching the training data.

**CutMix**: CutMix randomly cuts out a rectangular region from one image and pastes it into another image, while also mixing the corresponding labels.

Random Apply: Random Apply applies data augmentation techniques randomly to images, preventing the model from overfitting to specific augmentation patterns.

Training Strategies

YOLOv5 utilizes several training strategies to optimize the model's performance:

**Multi-scale training:** Trains the model on images resized to different scales, improving its ability to detect objects at different sizes.

**Adaptive learning rate**: Adjusts the learning rate dynamically during training to ensure optimal convergence.

**Early stopping**: Stops training when the model performance starts to plateau, preventing overfitting.

Ensemble learning: Combines multiple trained models to produce a more robust and accurate prediction.

**Additional Features**

YOLOv5 includes several additional features that enhance its capabilities:

**Non-maximum suppression** (NMS): Removes redundant bounding boxes that overlap significantly, ensuring that each detected object has a single, accurate bounding box.

**IoU** (Intersection over Union) loss: Measures the overlap between predicted bounding boxes and ground truth bounding boxes, guiding the model to predict more accurate bounding boxes.

**Generalized IoU** (GIoU) loss: An improved version of IoU loss that considers the shape of the bounding boxes, leading to more precise predictions.

YOLOv5's architecture, data augmentation techniques, training strategies, and additional features contribute to its impacts in detection, making it a popular choice for various applications.

### 1.2.3. YOLOv8 model

YOLOv8 is the latest version of the YOLO detection architecture, developed by Ultralytics. It builds upon the previous versions of YOLO, incorporating several significant enhancements to improve performance and efficiency. Model YOLOv8 exceeds all other object detection in precision and speed whose FPS is more than 200 and the accuracy is approximately 64.0% AP. Compared to its previous version, YOLOv8 is trained on COCO dataset which initially does not contain any pre-trained weights or biases.

The architecture of YOLOv8 comprises 3 primary components.

- **Backbone**: ELAN, E-ELAN
- **Neck**: SPPCSPC + PANet + RepConv
- **Head:** YOLOR and Auxiliary head

**Backbone**

The backbone of YOLOv8 is based on a modified version of the CSPDarknet53 architecture, which was introduced in YOLOv5. CSPDarknet53 is known for its high performance and efficiency, making it a suitable choice for object detection tasks. The backbone is created from the Stem Block which primarily used for extract features from the original input and pass those features to the ELAN demonstrated in **Figure 2.2**

**C2F Module**

YOLOv8 introduces a novel C2F module that replaces the traditional C3 module used in previous versions. The C2F module incorporates cross-stage feature fusion (CSF) and channel fusion (CF) techniques to improve feature extraction and reduce computational complexity.

**Decoupled Head**

YOLOv8 employs a decoupled head architecture, which separates the bounding box regression and classification branches. This approach allows for more efficient training and improved accuracy.

**Dynamic Architecture**

YOLOv8 employs a dynamic architecture that automatically adapts to the input image size. This feature enables the model to handle images of varying scales without compromising performance.

**Data Augmentation**

YOLOv8 utilizes a comprehensive set of data augmentation techniques, including mosaic augmentation, random flipping, and color jittering. These techniques help to improve the generalization ability of the model and reduce overfitting.

### Training Strategies

YOLOv8 implements several new training strategies, such as CIOU loss, GIOU loss, and PANet. These losses are designed to better handle irregular object shapes and improve the overall accuracy of the model.

### Additional Features

In addition to the architectural enhancements mentioned above, YOLOv8 also incorporates several other features that contribute to its performance:

Eliminate Grid Sensitivity: YOLOv8 eliminates grid sensitivity by introducing the Swish activation function and using adaptive anchors.

Balance Losses: YOLOv8 balances losses by assigning different weights to positive and negative samples.

Build Targets: YOLOv8 constructs targets using a novel algorithm that improves the efficiency and accuracy of the training process.

## 1.3. MEDIAPIPE

MediaPipe is an open-source framework developed by Google that provides a way to build pipelines for processing perceptual data, such as video and audio. It's primarily used for building machine learning pipelines to process and analyze multimedia data in real-time.

MediaPipe offers a set of pre-built components, tools, and machine learning models that developers can use to create applications for various tasks, including object detection, facial recognition, gesture recognition, pose estimation, augmented reality effects, and more.

It's known for its flexibility, allowing developers to combine and customize different modules within a pipeline to suit specific project needs. MediaPipe is often used in various domains like computer vision, augmented reality, robotics, and other fields requiring real-time multimedia processing. Figure 1 below describes a pipeline for an object detection process of MediaPipe.

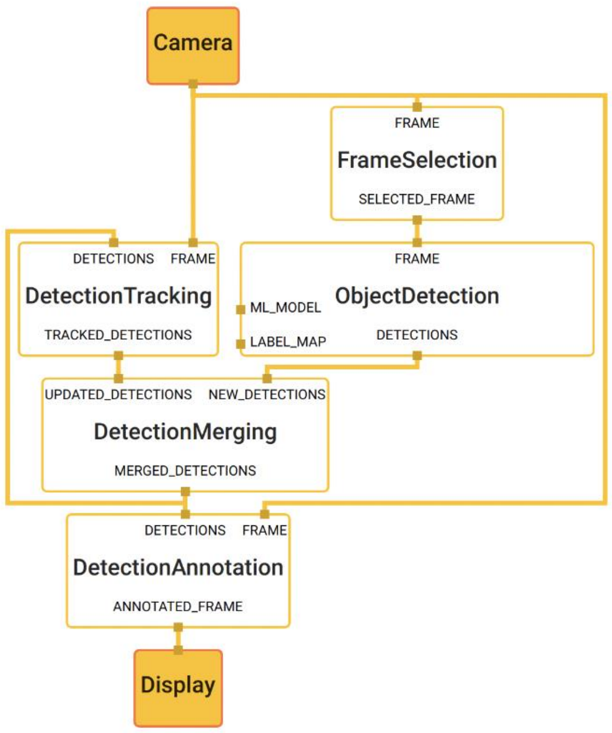


***Figure 9.*** *Object detection pipeline in MediaPipe [14]*

The main use case for MediaPipe is rapid prototyping of perception pipelines with inference models and other reusable components. MediaPipe also facilitates the deployment of perception technology into demos and applications on a wide variety of different hardware platforms. MediaPipe enables incremental improvements to perception pipelines through its rich configuration language and evaluation tools. MediaPipe allows a developer to prototype a pipeline incrementally. A pipeline is defined as a directed graph of components where each component is a *Calculator*. The graph is specified using a *GraphConfig protocol* buffer and then run using a Graph object.

In the graph, the calculators are by data, *Streams*. Each represents a time-series of data, *Packets*. Together, the calculators and streams define a data-flow graph. The packets which flow across the graph are collated by their timestamps within the time-series.

- *Packet*: basic data unit in MediaPipe. A packet consists of a numeric timestamp and a shared pointer to an immutable payload.
- *Stream*: each node in the graph is connected to another node through a stream.  A stream carries a sequence of packets whose timestamps must be monotonically increasing.

- ❖ *Calculators*: implemented as each node of a graph. The bulk of graph execution happens inside its calculator.
- ❖ *Graph*: all processing takes place within the context of a Graph. A graph contains a collection of nodes joined by directed connections along which packets can flow.
- ❖ *GraphConfig*: is a specification that describes the topology and functionality of a MediaPipe graph. All the necessary configurations of the node, such its type, inputs and outputs must be described in the specification. Description of the node can also include several optional fields, such as node-specific options, input policy and executor.

### 1.3.1. MEDIAPIPE POSE

#### 1.3.1.1. MediaPipe Pose pipeline

MediaPipe Pose is a ML solution for high-fidelity body pose tracking, inferring 33 3D landmarks and background segmentation masks on the whole body from RGB video frames utilizing our BlazePose research that also powers the ML Kit Pose Detection API. Current state-of-the-art approaches rely primarily on powerful desktop environments for inference, whereas our method achieves real-time performance on most modern mobile phones, desktops/laptops, in python and even on the web.

The solution utilizes a two-step detector-tracker ML pipeline, proven to be effective in MediaPipe Hands and MediaPipe Face Mesh solutions. Using a detector, the pipeline first locates the person/pose region-of-interest (ROI) within the frame. The tracker subsequently predicts the pose landmarks and segmentation mask within the ROI using the ROI-cropped frame as input. Note that for video use cases the detector is invoked only as needed, i.e., for the very first frame and when the tracker could no longer identify body pose presence in the previous frame. For other frames the pipeline simply derives the ROI from the previous frame’s pose landmarks.

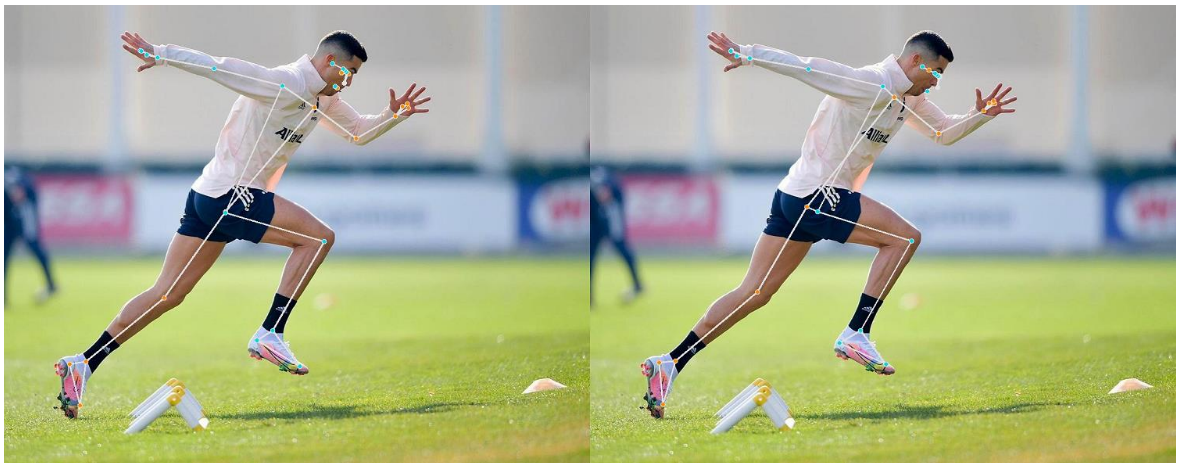


***Figure 10.*** *Example of pose detection by MediaPipe [15]*

### 1.3.1.2. BlazePose detector model

For real-time performance of the full ML pipeline consisting of pose detection and tracking models, each component must be very fast, using only a few milliseconds per frame. To accomplish this, it is known that the strongest signal to the neural network about the position of the torso is the person's face (due to its high-contrast features and comparably small variations in appearance). Therefore, BlazePose is a fast and lightweight pose detector by making the strong (yet for many mobile and web applications valid) assumption that the head should be visible for single-person use cases.

Therefore, the detector is inspired by the lightweight BlazeFace model, used in MediaPipe Face Detection, as a proxy for a person detector. It explicitly predicts two additional virtual keypoints that firmly describe the human body center, rotation and scale as a circle. Inspired by Leonardo’s Vitruvian man, the model predicts the midpoint of a person’s hips, the radius of a circle circumscribing the whole person, and the incline angle of the line connecting the shoulder and hip midpoints. Figure 3 below describes the detection’s pipeline of BlazePose detector model.

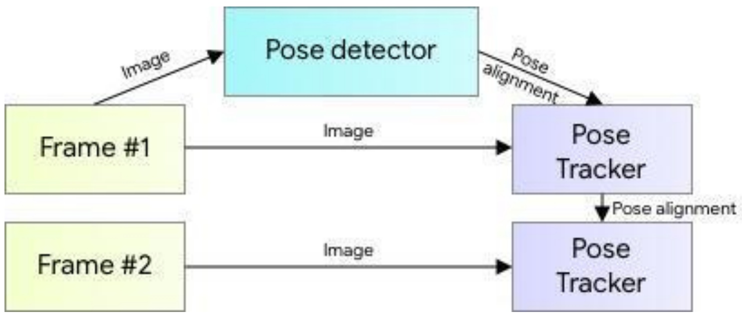


***Figure 11.*** *MediaPipe pose detection's pipeline [16]*

### 1.3.1.3. MediaPipe Pose Output

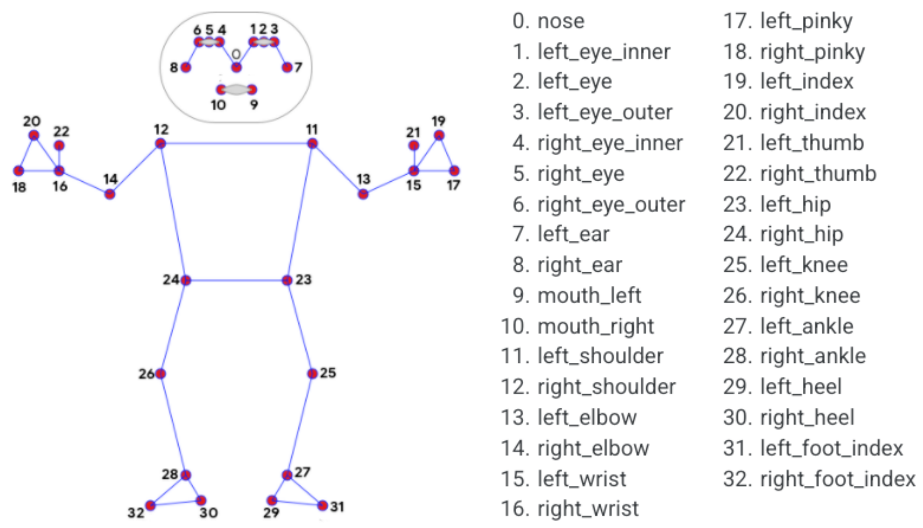


***Figure 12.*** *The 33 landmarks model in MediaPipe Pose predicts [17]*

MediaPipe Pose returns a list of 33 pose landmarks as shown from Figure 4 above. Each landmark consists of the following properties:

- ❖ *x* and *y*: Landmark coordinates normalized to *[0.0, 1.0]* by the image width and height respectively.
- ❖ *z:* Represents the landmark depth with the depth at the midpoint of hips being the origin, and the smaller the value the closer the landmark is to the camera. The magnitude of *z* uses roughly the same scale as *x*.
- ❖ *visibility*: A value in *[0.0, 1.0]* indicating the likelihood of the landmark being visible (present and not occluded) in the image.

## 1.4. RELATED TECHNOLOGIES AND LIBRARIES

### 1.4.1. Open CV

OpenCV is an open-source software library for computer vision and machine learning. The OpenCV full form is open-source computer vision library. It was created to provide a shared infrastructure for applications for computer vision and to speed up the use of machine perception in consumer products.

Gary Bradsky invented OpenCV in 1999 and soon the first release came in 2000. This library is based on optimized C/C++ and supports Java and Python along with C++ through an interface. The library has more than 2500 optimized algorithms, including an extensive collection of computer vision and machine learning algorithms, both classic and state-of-the-art. Using OpenCV it becomes easy to do complex tasks such as identify and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D object models, generate 3D point clouds from stereo cameras, stitch images together to generate an entire scene with a high-resolution image and many more.

### 1.4.2. Scikit-learn

Scikit-learn is a free software library for the Python programming language that provides a collection of algorithms for machine learning and data mining. It features various classification, regression and clustering algorithms including support vector machines, random forests, boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. It is licensed under the BSD license.

### 1.4.3. Keras

Keras is an open-source deep learning framework for python. It has been developed by an artificial intelligence researcher at Google named Francois Chollet. Leading organizations like Google, Square, Netflix, Huawei and Uber are currently using Keras. Keras has nice guiding principles: modularity, minimalism, extensibility, and Python-nativeness. Keras also has out-of-the-box implementations of common network structures. It’s fast and easy to get a convolutional neural network up and running.

# CHAPTER 2. METHODOLOGY

## 2.1. VIOLENCE ACTION SELECTION

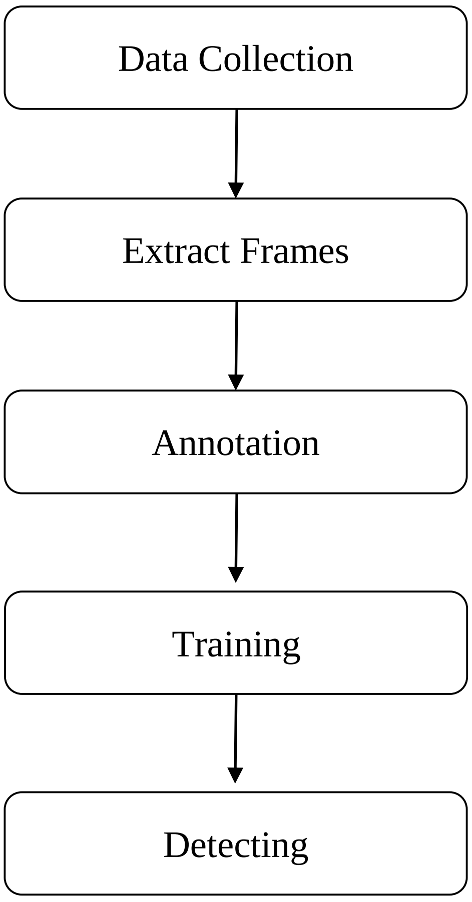
First and foremost, the initial step of this thesis is to decide which actions to use to train the machine learning model for children violence detection. Since the time for this work is limited, only 3 actions are selected. Based on VeryWellMind about the most common physical children violence actions, I decided on 4 actions: *pinching, kicking, slapping, and choking.*

***Figure 13.*** *An adult slapping a child [18]*

After deciding on 4 actions, for each action, the next step is to identify what are the posture of those violence actions. Here are the identified common posture for each actions:

- ❖ *Pinching*: Loose upper arm, weak peak contraction and lean-back standing posture.
- ❖ *Choking*: Lower back and High back.
- ❖ *Slapping*: Foot placement too tight/wide and knee placement too wide/tight.
- ❖ *Kicking*: Knee' angles and knee over toe while going down.

The overall approach of YOLOv8 and the MediaPipe framework is similar. Both of them include some very fundamental steps:



## 2.2. YOLOv8

### 2.2.1. Development Environment

Using Google Colaboratory which is a product of Google support Python code, evaluate models.

Using support libraries such as OpenCV, numpy.

Using Tensorflow library supports computing accuracy and other evaluation values while training the model.

**Concepts of parameters in training model:**

- Input: the input of model (**imgsz = 640** meaning that all the input images are resized into 640x640 before training).
- Batch-size: Dataset is too tremendous, thus, it should be separated in batches before training. This helps avoid memory overflow.
- Decay: technique to reduce the sophistication of model
- Learning-rate: learning rate of model using in Gradient Descent, model commence with low learning rate and then increase according to the initialized number.

- Step train (Epoch): The number of iterations of model through datasets.
- Number of class: Number of labels that have been defined.
- Ignore_thresh: this value range between 0 and 1, if any bounding box with confidence score below this number will be eliminated.

### 2.2.2. Collecting Data

Due to the lack of videos or records on the internet of adults hitting children with high quality. Most of the videos are sneakily recorded or urgently. This makes the model unable to properly annotate key features in the input images. The majority of the self-collected videos were either taken by myself, sisters, my niece, or my family. There are a total of 2 contributors for this data set. For each action, each participant must:

❖ Recorded video must be clear and actions should be performed at appropriate speed (not too fast or too slow)

For any video provided by a contributor, the video must meet the following criteria:

❖ The video is shot in an environment with sufficient light so that a person's entire body can be easily seen.
❖ For each action, the video must clearly show the important joints or movements that are important to the action.
❖ The participants must be in the frame throughout the video.

After collecting the videos, videos are extracted into frames using OpenCV libraries. The data is suggested to be rename in order so that it can be easily managed for future usage.

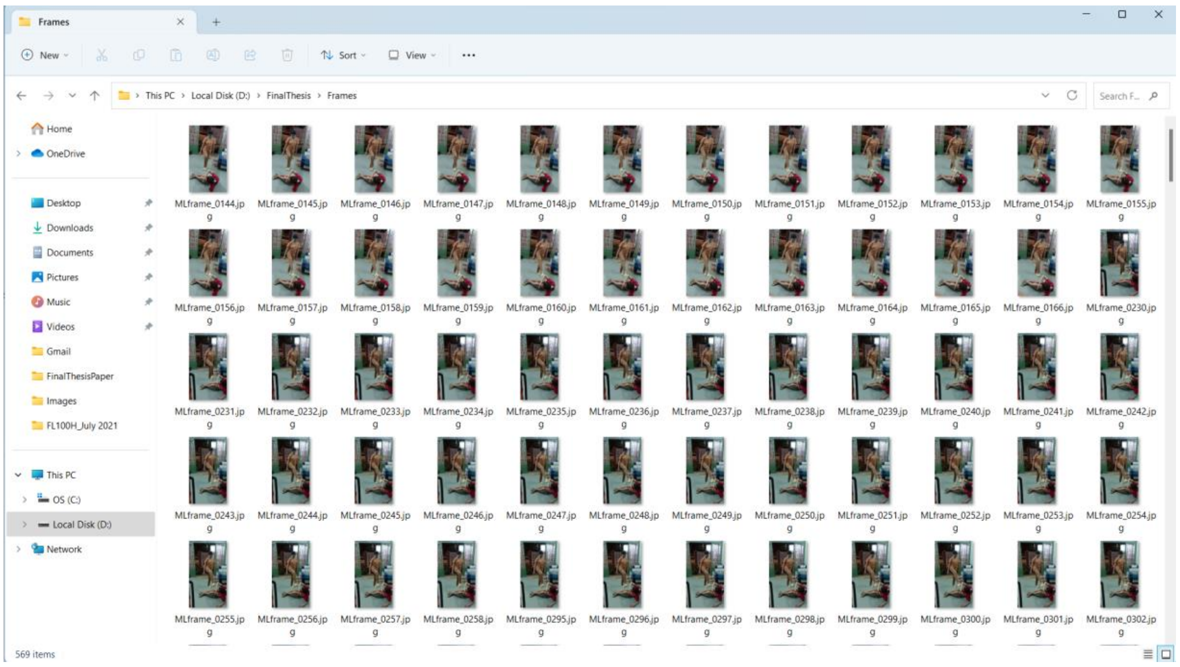


*__Figure 14.__ Extracted Image from Video*

In the collected data, number of images in every action namely *pinching*, *choking*, *kicking*, *slapping* and *normal* are shown in **Table 1**

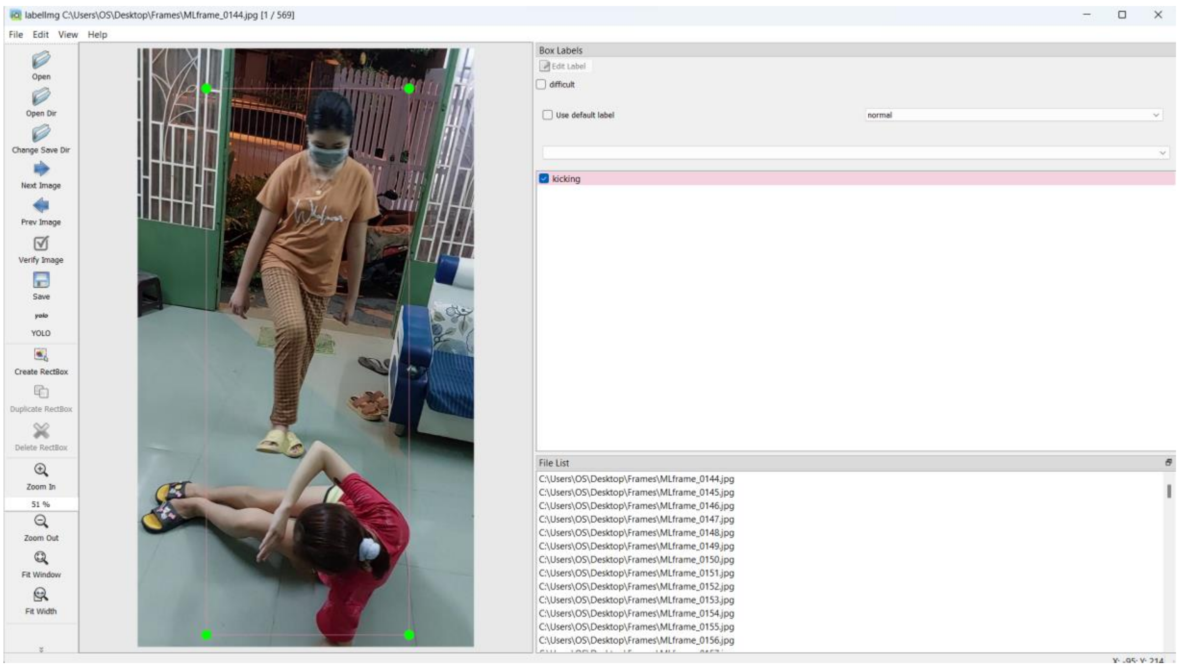
*__Table 1.__ Description of images*

| **Types of image** | **Number** |
|---|---|
| Slapping | 1814 |
| Kicking | 2755 |
| Choking | 3692 |
| Pinching | 1045 |
| Normal | 3351 |
| **Sum** | **12657** |

In this dataset, an augmentation approach, called reflecting images, has been used to enlarge the number of images.

Original Image Reflected Image

Using tool such as LabelImg demonstrated in figure 12 to annotate manually for 12657 collected images. This thesis primarily concentrates specifically on detecting 4 violence classes. Therefore, 4 classes are called “*pinching”, “choking”, “slapping”*, and “*kicking”.*

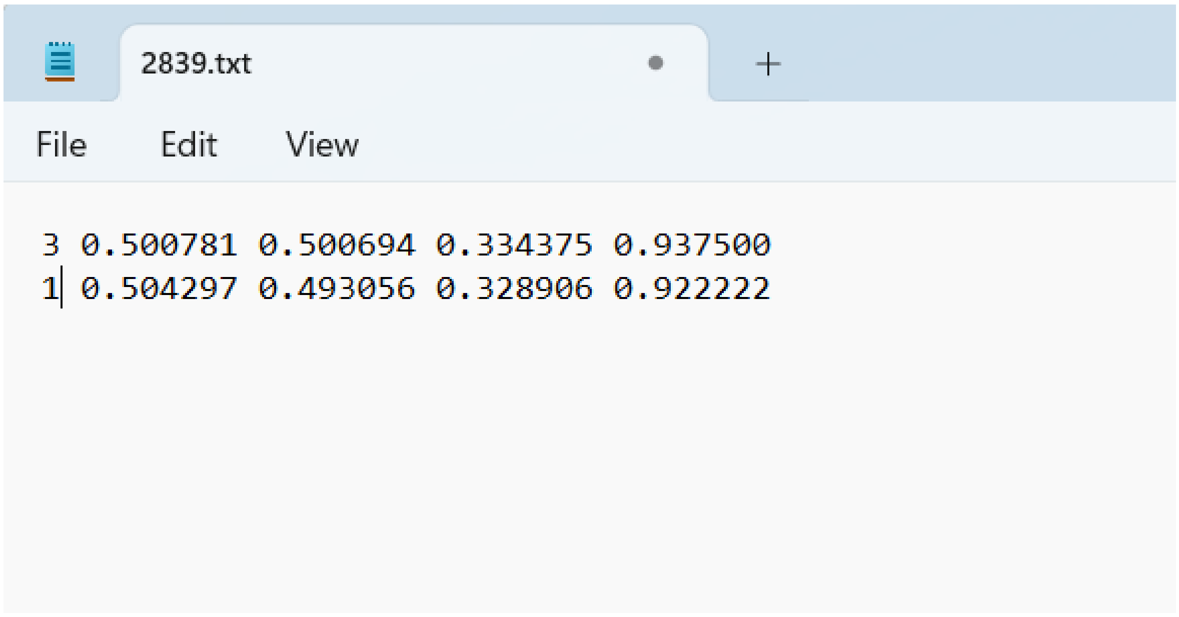


```
labelImg C:\Users\OS\Desktop\Frames\MLframe_0144.jpg [1 / 569]
```

```
C:\Users\OS\Desktop\Frames\MLframe_0144.jpg
C:\Users\OS\Desktop\Frames\MLframe_0145.jpg
C:\Users\OS\Desktop\Frames\MLframe_0146.jpg
C:\Users\OS\Desktop\Frames\MLframe_0147.jpg
C:\Users\OS\Desktop\Frames\MLframe_0148.jpg
C:\Users\OS\Desktop\Frames\MLframe_0149.jpg
C:\Users\OS\Desktop\Frames\MLframe_0150.jpg
C:\Users\OS\Desktop\Frames\MLframe_0151.jpg
C:\Users\OS\Desktop\Frames\MLframe_0152.jpg
C:\Users\OS\Desktop\Frames\MLframe_0153.jpg
C:\Users\OS\Desktop\Frames\MLframe_0154.jpg
C:\Users\OS\Desktop\Frames\MLframe_0155.jpg
C:\Users\OS\Desktop\Frames\MLframe_0156.jpg
```

***Figure 15.*** *LabelImg supports annotation*

In object detecting models, every model uses labels in despair extensions. Specifically, YOLO uses files of extension *.txt*.



```
3 0.500781 0.500694 0.334375 0.937500
1 0.504297 0.493056 0.328906 0.922222
```

***Figure 16.*** *Example of YOLO annotation*

After annotating an object in the YOLO file, the label is created in **Figure 13**, every line including the name of the label and the coordinates of the object:

<**object-class**><**x**><**y**><**width**><**height**>

**<object-class>** is the label of object encoded into 0-4 (0 is for *normal*, 1 is *slapping*, 2 is *choking*, 3 is *kicking*, 4 is *pinching*).

<**x**><**y**><**width**><**height**> respectively coordinates of center and size of height and width of object. Those values are fine tuning and range in [0,1].

After the data has been annotated, the next step is to set up the model.

### 2.2.3. Set up the model YOLOv5

#### Dividing the dataset

Using Hold-up technique to divide the model in 2 separate components in which 80% is for the training set and 20% is for the validation set. The workflow of division is demonstrated in **Figure 15**.

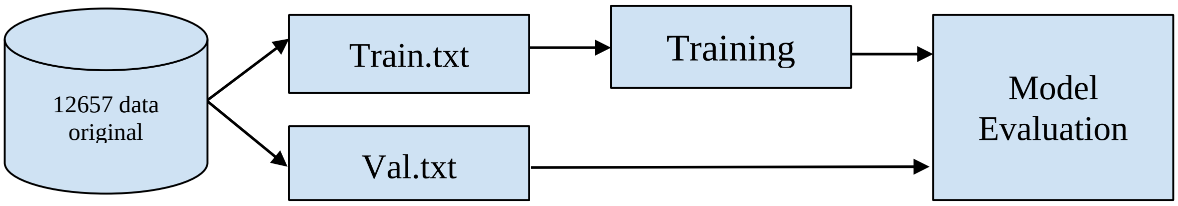


***Figure 17.*** *Demonstration of training process*

***Table 2.*** *Configuration of hyperparameters in YOLOv5 models*

| **Description of hyper-parameters** | | | | | |
|---|---|---|---|---|---|
| **No.** | **Type** | **Value** | **No.** | **Type** | **Values** |
| **1** | **Input-image** | 640x640 | **5** | **Number of classes** | 5 |
| **2** | **Batch size** | 8 | **6** | **Decay** | 0.0005 |
| **3** | **Subdivision** | 16 | **7** | **Momentum** | 0.937 |
| **4** | **Learning rate** | 0.01 | **8** | **Num-step** | 128 |

### 2.2.3. Set up the model YOLOv8

**Dividing the dataset**

Dividing dataset in YOLOv8 is similar to YOLOv5.

**Configuration of hyperparameters in YOLOv8 models**

***Table 3.*** *Describe hyper-parameters in training YOLOv8*

| **Description of hyper-parameters** | | | | | |
|---|---|---|---|---|---|
| **No.** | **Type** | **Value** | **No.** | **Type** | **Values** |
| **1** | **Input-image** | 640x640 | **5** | **Number of classes** | 5 |
| **2** | **Batch size** | 8 | **6** | **Decay** | 0.0005 |
| **3** | **Subdivision** | 32 | **7** | **Momentum** | 0.937 |
| **4** | **Learning rate** | 0.01 | **8** | **Num-step** | 6000 |

## 2.3. VIOLENCE DETECTION USING MEDIAPIPE

### 2.3.1. Data processing

The general approach of processing data from collected videos for training model is illustrated in **Figure 7**: Data processing below.

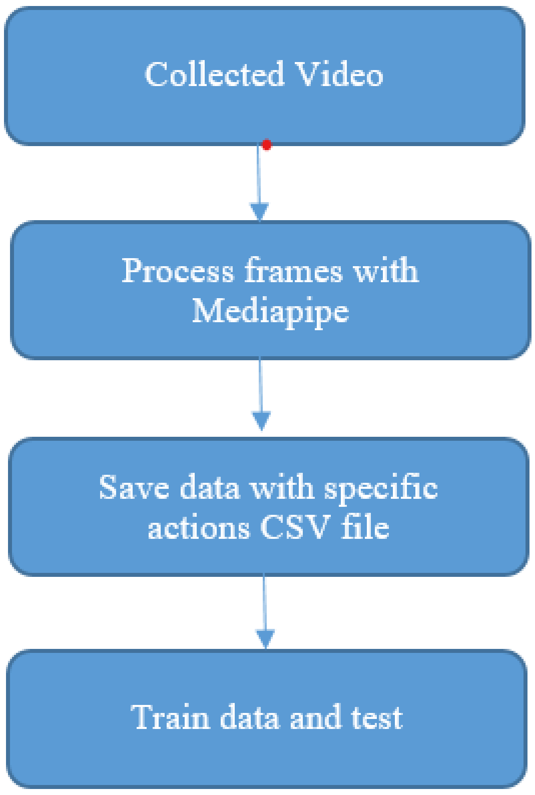

Collected Video
Process frames with Mediapipe
Save data with specific actions CSV file
Train data and test

***Figure 18.*** *Data processing*

### 2.3.2. Extract data from video to file using OpenCV and MediaPipe

With every action, there are videos which are separated into different classes. With each video, every frame that matches a certain form of an action would be processed as follows:

- ❖ Process every frame using OpenCV then utilize MediaPipe Pose to produce a list of coordinate predictions of all keypoint locations, and their corresponding prediction confidence.

Configuration options for Mediapipe Pose:

- ➢ `MIN_DETECTION_CONFIDENCE`: Minimum confidence value `([0.0, 1.0])` from the person-detection model for the detection to be considered successful. Default to `0.5`.
- ➢ `MIN_TRACKING_CONFIDENCE`: Minimum confidence value `([0.0, 1.0])` from the landmark-tracking model for the pose landmarks to be considered tracked successfully, or otherwise person detection will be invoked automatically on the next input image. Setting it to a higher value can increase robustness of the solution, at the expense of a higher latency. Default to `0.5`.

For every frame with the list of predicted landmarks, 33 landmarks for a corresponding action would be extracted in append as a row to the csv file with the label column matching the class of the video.

- ❖ The collected data which is saved in a csv file would be split to train and evaluate model. 80% of the data will be used for model training and the remaining 20% will be used for model evaluation.

### 2.3.4. Classification with Scikit-learn

The machine learning algorithms used for model training are: Decision Tree/Random Forest (RF), K-Nearest Neighbors (KNN), C-Support Vector (SVC), Logistic Regression classifier (LR) and Stochastic Gradient Descent classifier (SGDC). After the training session, each result from each algorithm will be evaluated with metrics such as: precision, recall, accuracy and F1 score. The algorithm which provides the best results will be selected. Below is the sample code for this section.

```
algorithms = [("LR", LogisticRegression()),
    ("SVC", SVC()),
    ('KNN, KNeighborsClassifier()),
    ("SGDC", SGDClassifier()),
    ('RF', RandomForestClassifier()),]
for name, model in algorithms:
    trained_model = model.fit(X_train, y_train)
```

## 2.4. VIOLENCE DETECTION PROCESS IN DEPTH FOR EACH ACTION

The previous section discusses the general approach, each step to build models for the 4 actions, this section will address in detail on each action.

### 2.4.1. Description of some violent actions

-*Slapping*: slapping children refers to the act of physically hitting a child with an open hand, usually on the buttocks, legs, or hands, as a form of discipline or punishment. The action and either be performed while standing or sitting, involves punishers raising their hands and slapping children. The model is created to detect the raising hand of adults, warning about feasible slapping action.

-*Kicking*: Kicking children in this research is described as an action involving physically striking a child with a kick, usually directed at their body, limbs, or other parts.

-*Choking*: Choking children is an action where an adult using both hands chokes children' neck which causes them difficulties in breathing.

-*Normal*: Normal activities involve action namely playing with children, having fun, taking part in many daily activities.

## 2.5. Results

### 2.5.1. Slapping

Figure 9 is a graph that visualizes the number of frames from videos. There are total of 3243 images which are divided into 2 sets of normal (1172 images) and slapping (2071 images).

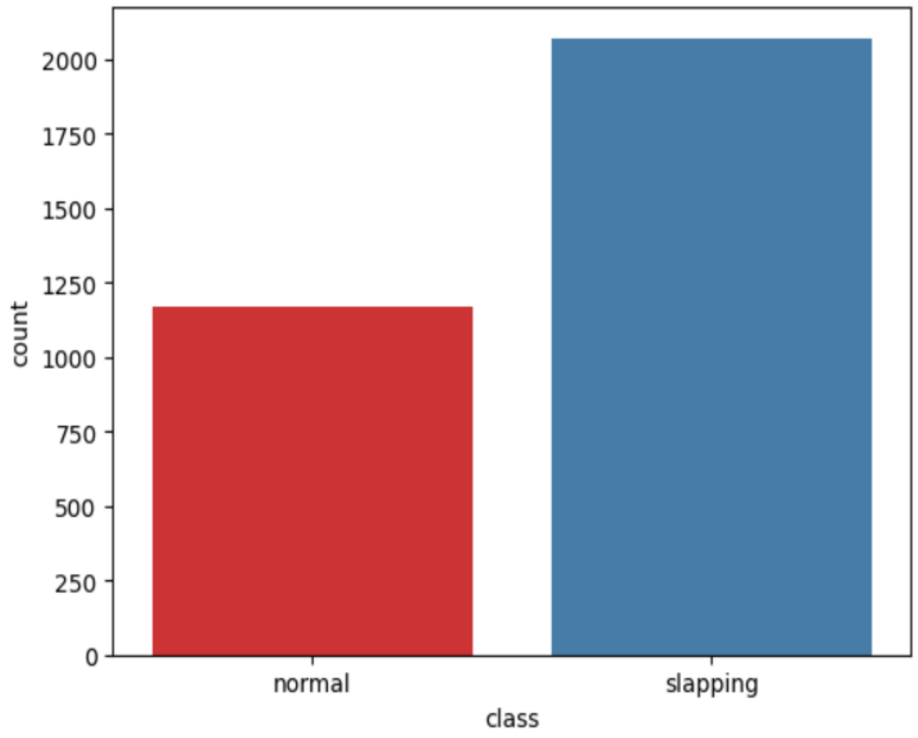


***Figure 19.*** *Class balance of Slapping's dataset*

The results of the experiment in training mentioned model are shown in the table below.

***Table 4.*** *Model training experiments for Slapping Action*

| Model | Precision | Recall | Accuracy |
|---|---|---|---|
| LR | 0.967 | 0.964 | 0.958 |
| RC | 0.967 | 0.964 | 0.958 |
| RF | 0.985 | 0.976 | 0.988 |
| GB | 0.982 | 0.982 | 0.976 |
| NB | 0.992 | 0.983 | 0.985 |

### 2.5.2 Kicking

Figure 9 is a graph that visualizes the number of frames from videos. There are total of 2193 images which are divided into 2 sets of normal (1127 images) and slapping (1066 images).

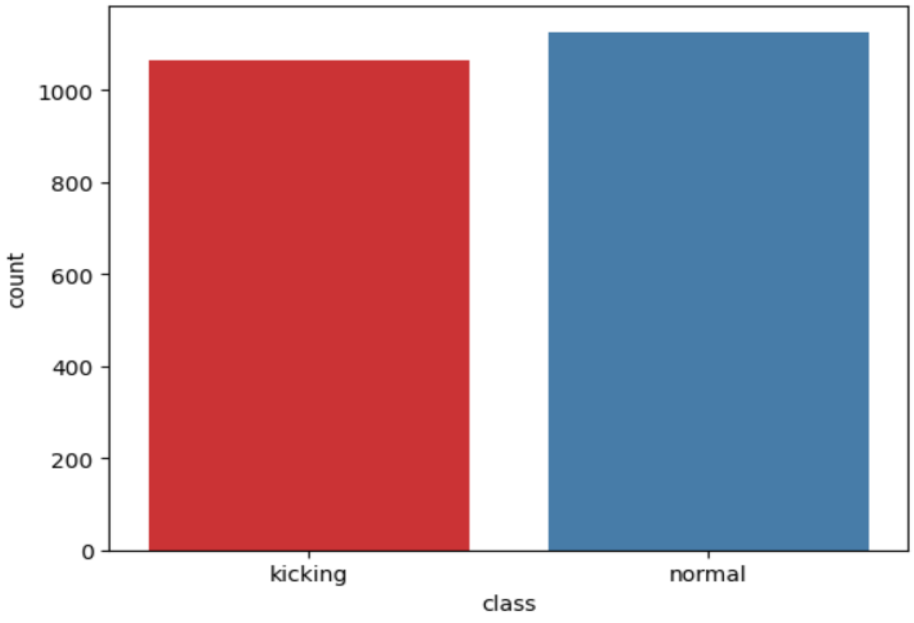


***Figure 20.*** *Class balance of Kicking's dataset*

The results of the experiment in training mentioned model are shown in the table below.

***Table 5.*** *Model training experiments for Kicking Action*

| Model | Precision | Recall | Accuracy |
|---|---|---|---|
| LR | 0.983 | 0.98 | 0.984 |
| SVC | 0.992 | 0.91 | 0.995 |
| KNN | 0.974 | 0.975 | 0.912 |
| DTC | 0967 | 0.971 | 0.981 |
| NB | 0.913 | 0.923 | 0.925 |

### 2.5.3 Choking

Figure 9 is a graph that visualizes the number of frames from videos. There are total of 3243 images which are divided into 2 sets of normal (1712 images) and slapping (1513 images).

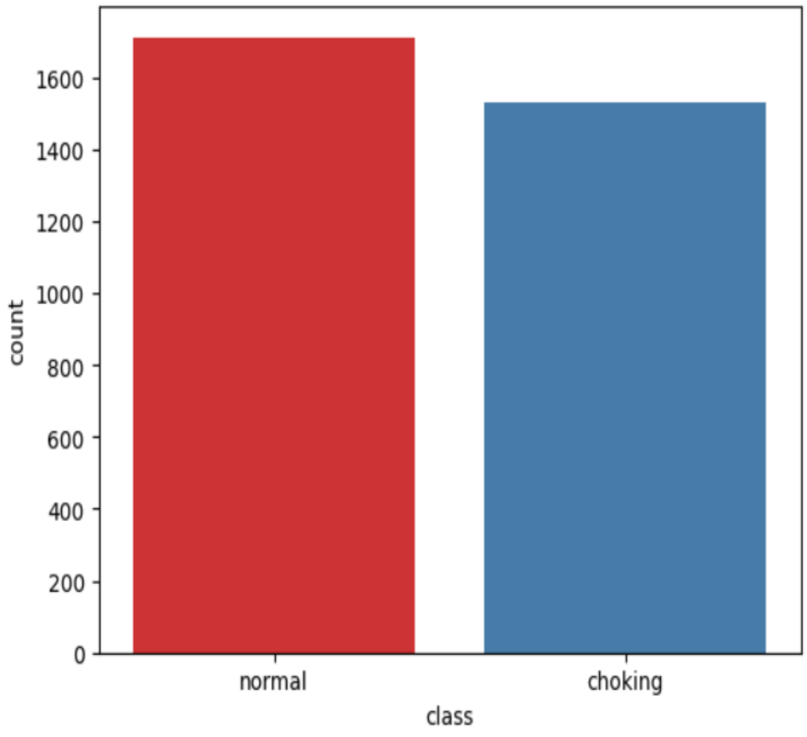


***Figure 21.*** *Class balance of Choking's dataset*

The results of the experiment in training mentioned model are shown in the table below.

***Table 6.*** *Model training experiments for Choking Action*

| Model | Precision | Recall | Accuracy |
|---|---|---|---|
| LR | 0.961 | 0.961 | 0.961 |
| SVC | 0.87 | 0.827 | 0.827 |
| KNN | 0.751 | 0.529 | 0.529 |
| DTC | 0.766 | 0.581 | 0.581 |
| NB | 0.836 | 0.791 | 0.791 |

# CHAPTER 3. RESULTS

## 3.1. MODEL ASSESSMENT

### 3.1.1. Evaluation criteria

There are some criteria in order to evaluate the quality of Machine Learning model and they are considered as the most essential part of our project. Otherwise, model will be meaningless. Some of those following criteria are common to assess the model.

- ❖ *Precision*: Precision is a measure of how reliable a positive prediction is. A high precision score means that most of the positive predictions made by the model are actually correct. A low precision score means that the model is making a lot of false positives. Precision is calculated by the following formula:
- ❖ *ROC curve (receiver operating characteristic curve)*: a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate (TPR) and False Positive Rate (FPR).
- ❖ *Confusion matrix*: a confusion matrix is a table that is used to evaluate the performance of a classification model. It is a square matrix that summarizes the model's predictions and the actual labels of the data. The rows of the matrix represent the actual labels of the data, and the columns of the matrix represent the model's predictions.
- ❖ *Recall*: tells the proportion that the model is accurately classifying the true positives. It is also called Sensitivity.
- ❖ *F1 score*: defined as the harmonic mean of precision and recall. The higher the precision and recall, the higher the F1-score.

## 3.2. Evaluation results

### 3.2.1 YOLO5 model

**Result of training YOLOv5 model with different versions:**

The result of training 3 despair YOLOv5 models described in **Table 3**:

***Table 7.*** *Results of 3 despair YOLOv5 models*

| Name | IoU | Precision | Recall | F1-score | mAP |
|---|---|---|---|---|---|
| YOLOv5.n | 0.572 | 0.926 | 0.844 | 0.97 | 0.98 |
| YOLOv5.s | 0.543 | 0.942 | 0.99 | 0.97 | 1 |
| YOLOv5.m | 0.651 | 0.81 | 0.851 | 0.87 | 0.98 |

After training 3 versions of YOLOv5, the optimal version is **YOLOv5.s**, and the graph evaluating loss of the YOLOv5.s is demonstrated in **Figure 16**

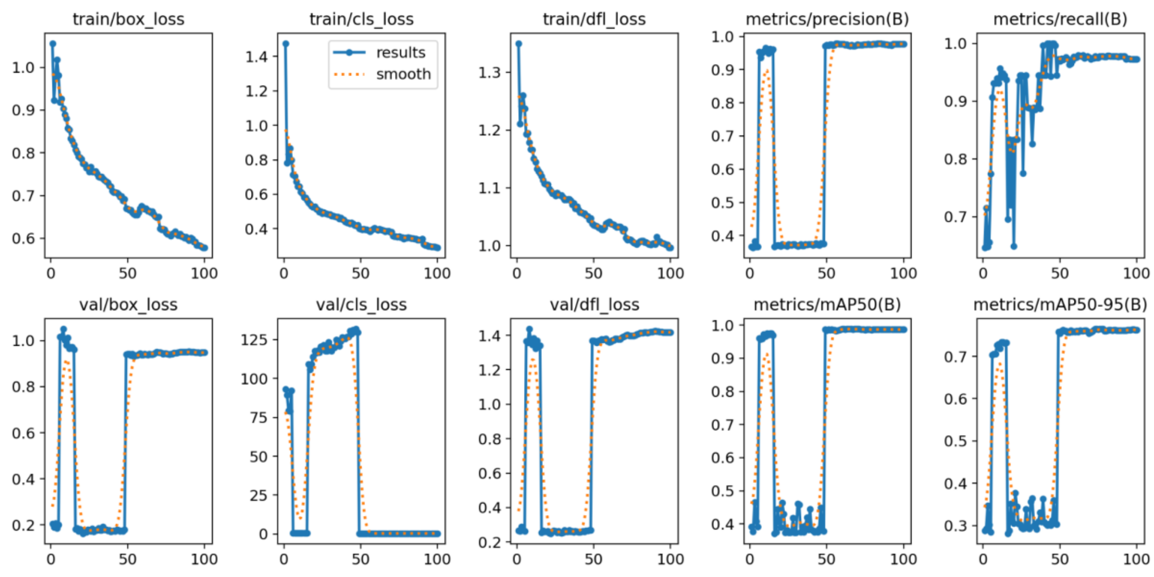


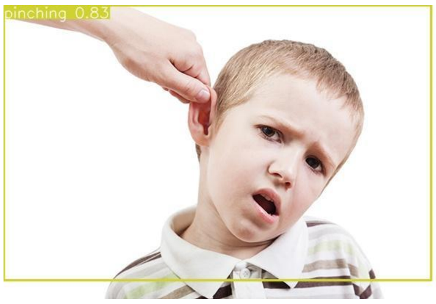
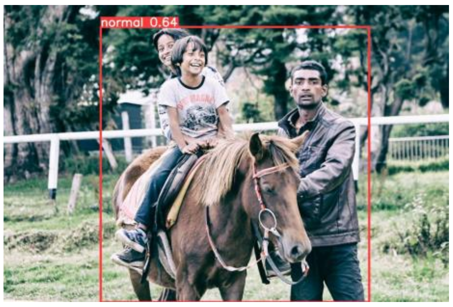
***Figure 22.*** *Loss graph after training and evaluate model YOLOv5s*

The loss function YOLOv5 is evaluated based on the training and validation sets after 100 epochs. According to **Figure 3.10**, the convergence of Box, Objectness, and Classification is dramatically improved after 100 epochs.

## Experiment on YOLOv5

When the model finishes training, model YOLOv5.s has the most optimized result. And **Table 4** is the result of the detection.

***Table 8.*** *Result of YOLOv5.s detection*

| Input | Output | Precision | Comment |
|---|---|---|---|
| Input: Image. Size: 1080x1920. Target: detect a violent action of adult performs on a child |  | Pinching: 83% | Model performs well and there is no mistake between actions. |
| Input: Video. Size 1080x1920. Target: detect a violent action of an adult performs on a child. |  | Normal: 64% | Model performs well and there is no mistake in detecting kicking actions. |

**Result of training YOLOv8 model with different versions:**

The result of training 3 despair YOLOv8 models described in **Table 6**:

***Table 9.*** *Results of 3 despair YOLOv8 models*

| Name | IoU | Precision | Recall | F1-score | mAP |
|---|---|---|---|---|---|
| YOLOv8.n | 0.571 | 0.954 | 0.844 | 0.97 | 0.975 |
| YOLOv8.s | 0.651 | 0.81 | 0.851 | 0.87 | 0.925 |
| YOLOv8.m | 0.52 | 0.954 | 1 | 0.98 | 0.98 |

**Figure 16** describes the mAP and loss score in YOLOv8.m created while training. This figure shows converging of model throughout the process.

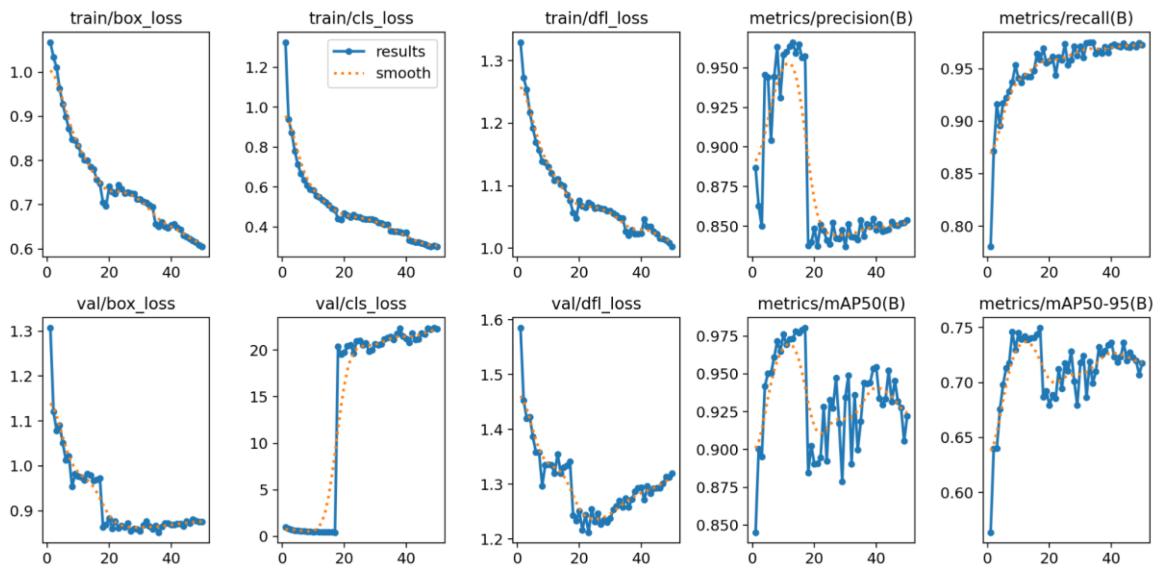
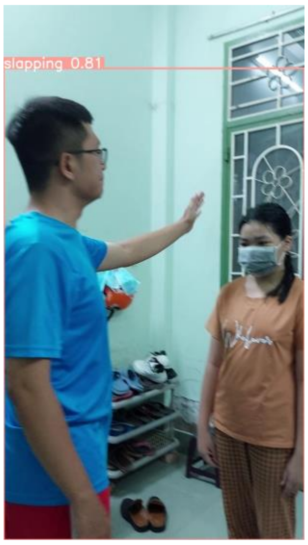
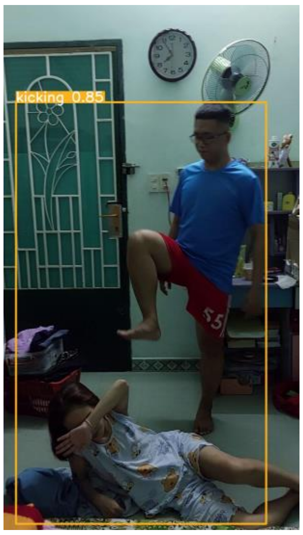


***Figure 23.*** *Loss graph after training and evaluate model YOLOv8*

## Experiment on YOLOv8

When the model finishes training, model YOLOv8.m has the most optimized result. And **Table 7** is the result of the detection.

***Table 10.*** *Result of YOLOv8.m detection*

| Input | Output | Precision | Comment |
|---|---|---|---|
| Input: Image. Size: 1080x1920. Target: detect a violent action of adult performs on a child |  | Slapping: 81% | Model performed well and there was no mistake between actions. |
| Input: Video. Size 1080x1920. Target: detect a violent action of an adult performs on a child. |  | Kicking: 85% | Model performed well and there was no mistake in detecting kicking actions. |

Using 50 images from the recorded video by extracting frames, in a minimal noise environment, the distance between cameras and violent actions is about 2 meters. First, we annotate 50 images (ground-truth). There are 40 images containing violent actions (10 for each), the rest are normal. Secondly, we test the set and evaluate the best model which are YOLOv8m and YOLOv5s respectively. The results are shown below.

***Table 11.*** *Confusion matrix of YOLOv8m*

| | Slapping | Kicking | Choking | Pinching | Normal |
|---|---|---|---|---|---|
| Slapping | 10 | 0 | 0 | 1 | 0 |
| Kicking | 0 | 9 | 0 | 0 | 0 |
| Choking | 0 | 0 | 6 | 0 | 0 |
| Pinching | 0 | 0 | 2 | 7 | 0 |
| Normal | 0 | 2 | 2 | 1 | 10 |

***Table 12.*** *Confusion matrix of YOLOv5s*

| | Slapping | Kicking | Choking | Pinching | Normal |
|---|---|---|---|---|---|
| Slapping | 8 | 1 | 0 | 5 | 0 |
| Kicking | 4 | 8 | 0 | 0 | 0 |
| Choking | 3 | 0 | 6 | 1 | 0 |
| Pinching | 0 | 0 | 2 | 5 | 0 |
| Normal | 0 | 1 | 2 | 3 | 10 |

### 3.2.2. Mediapipe and Skitlearn Algorithm

**Slapping**

Base on the metrics in table , the best model for this type of action is Gaussian Naive Bayes (NB) and the model give the following result

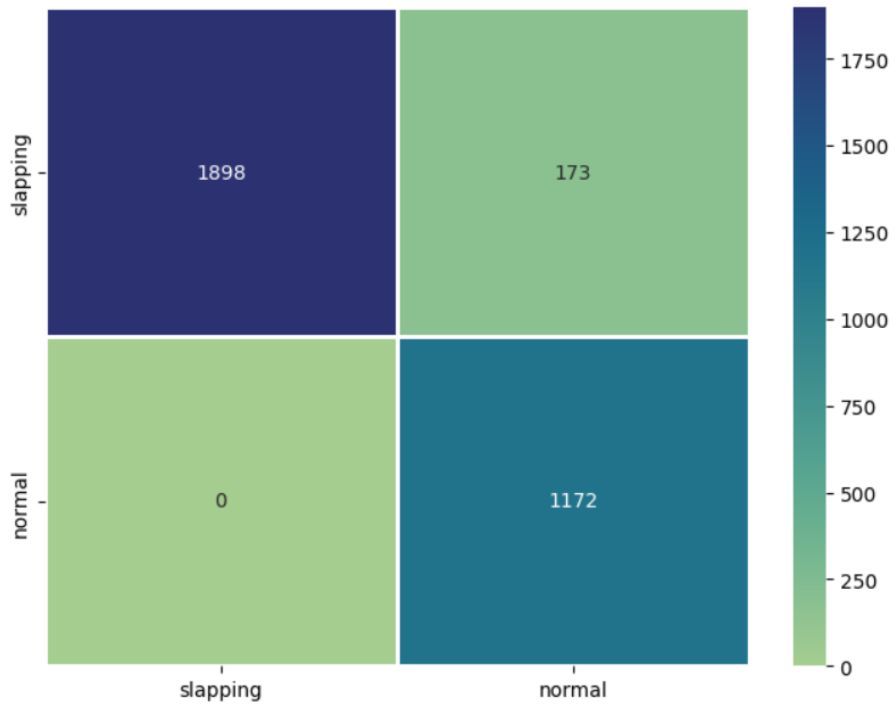


***Figure 24.*** *Confusion matrix Slapping action*

The following figures are F1 curve and ROC curve.

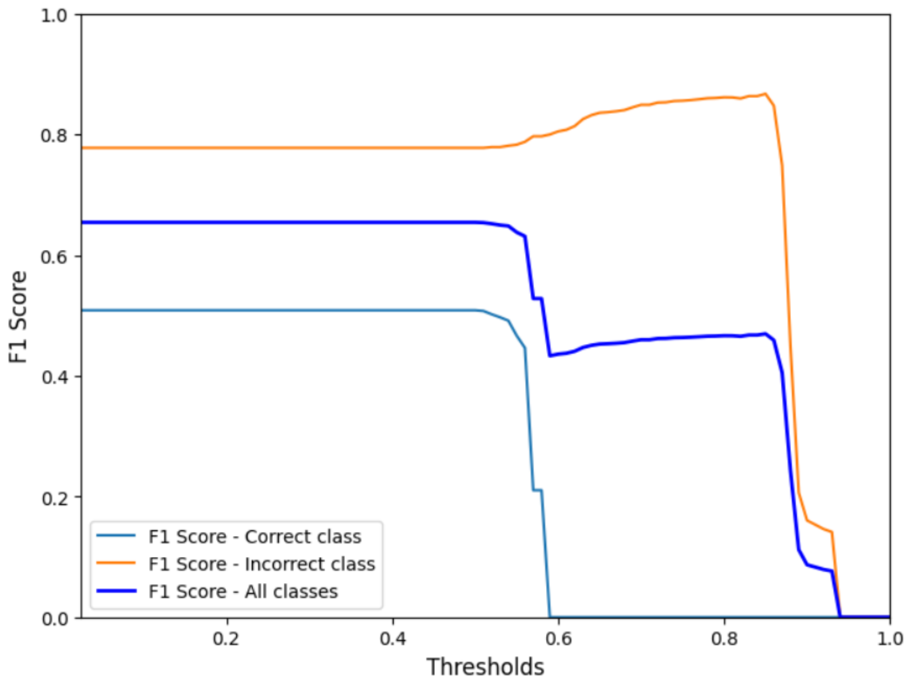


***Figure 25.*** *F1 curve of Slapping action*

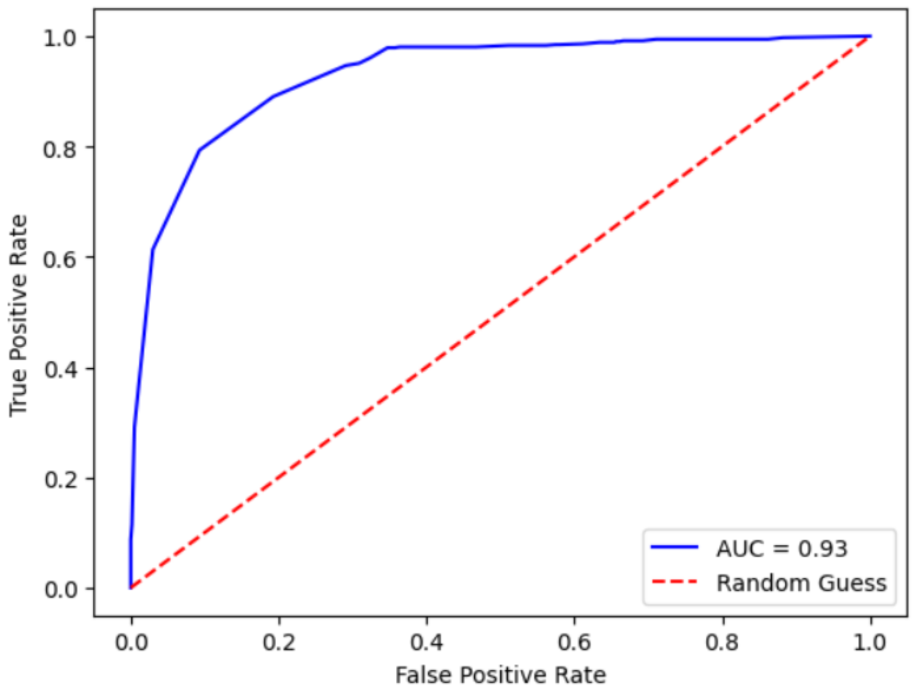


***Figure 26.*** *ROC curve of Slapping action*

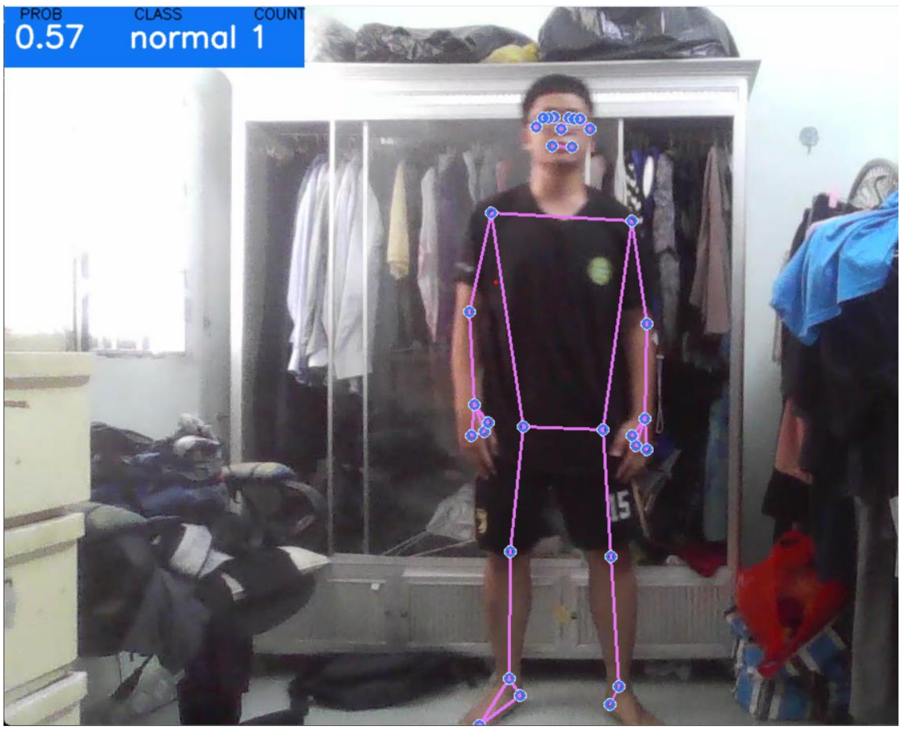


***Figure 27.*** *A person is in normal action*

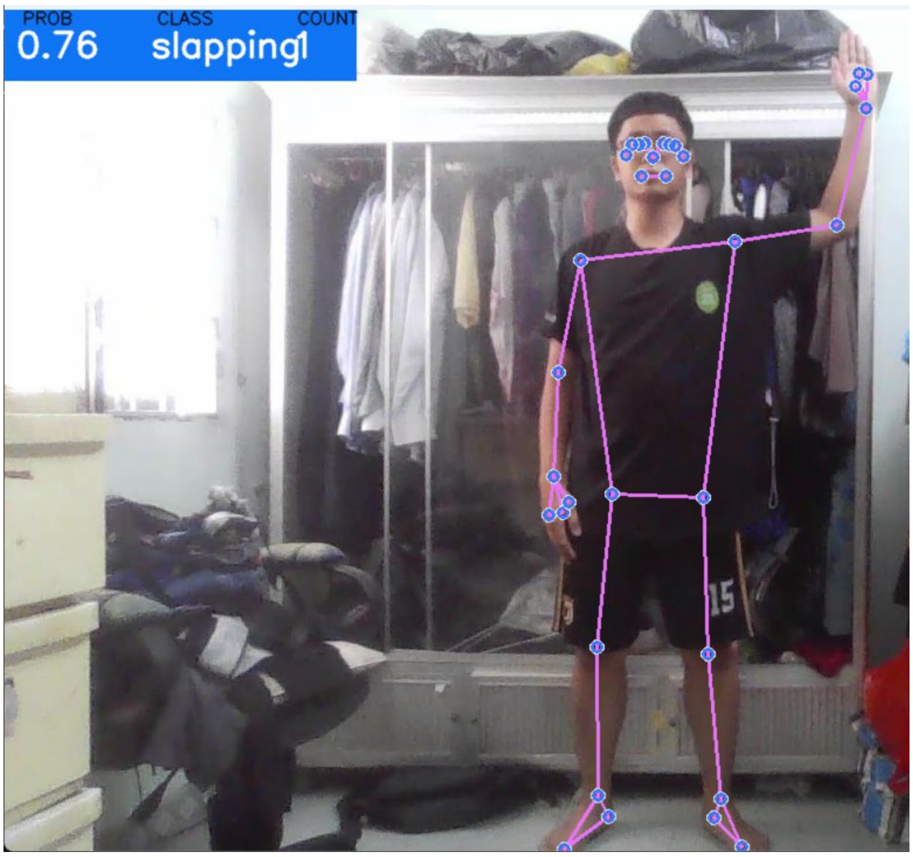


***Figure 28.*** *A person is in the position of violence action*

**Kicking**

Base on the metrics in **Table 9**, the best model for this type of action is Gaussian Naive Bayes (NB) and the model give the following result

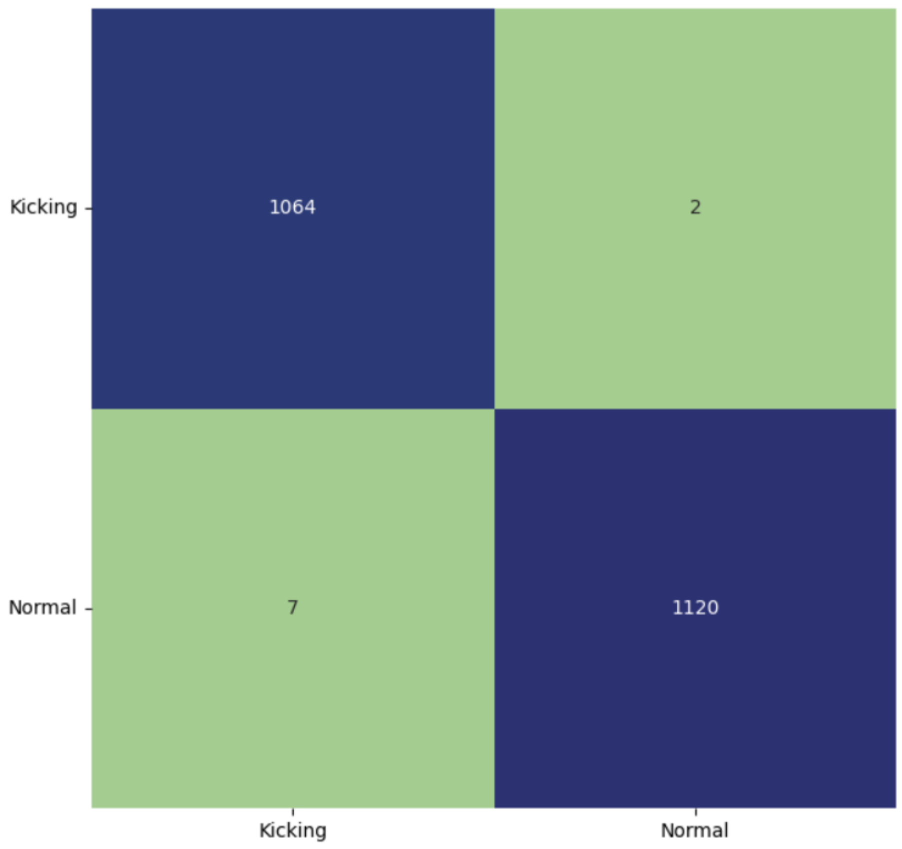


***Figure 29.*** *Confusion matrix of Kicking action*

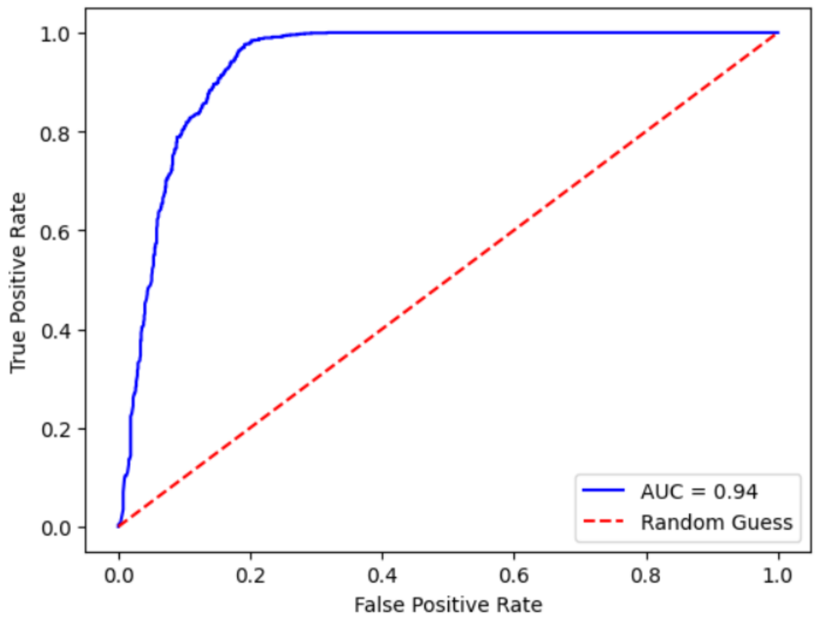


***Figure 30.*** *ROC curve of Kicking action*

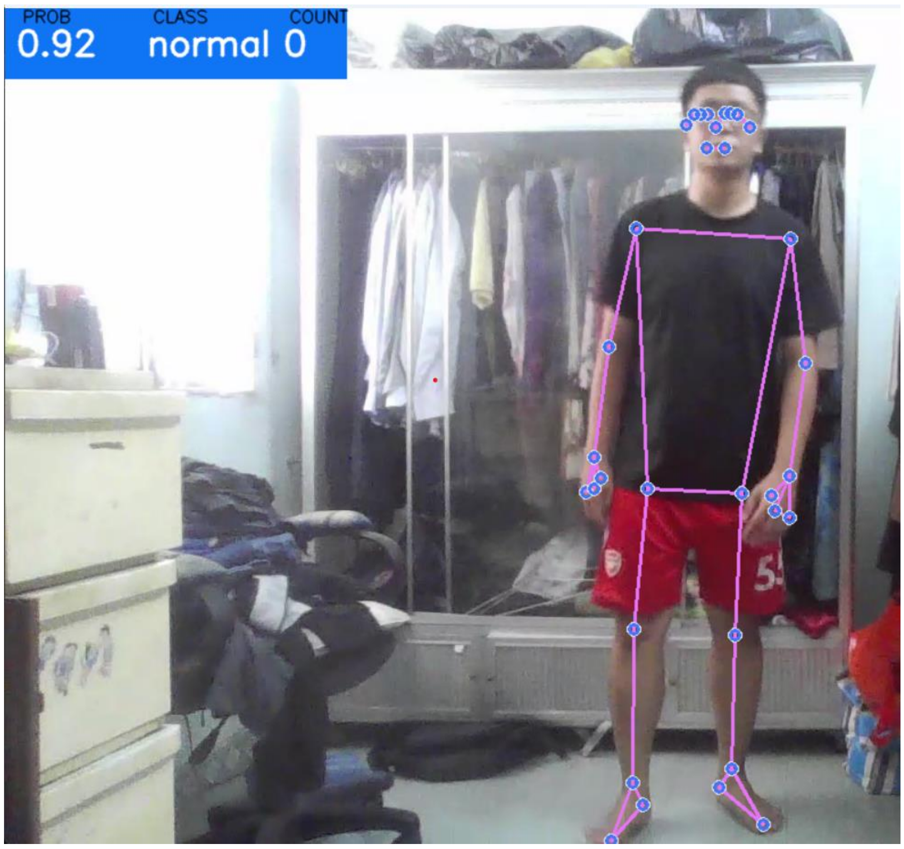


***Figure 31.*** *A person is standing normally*

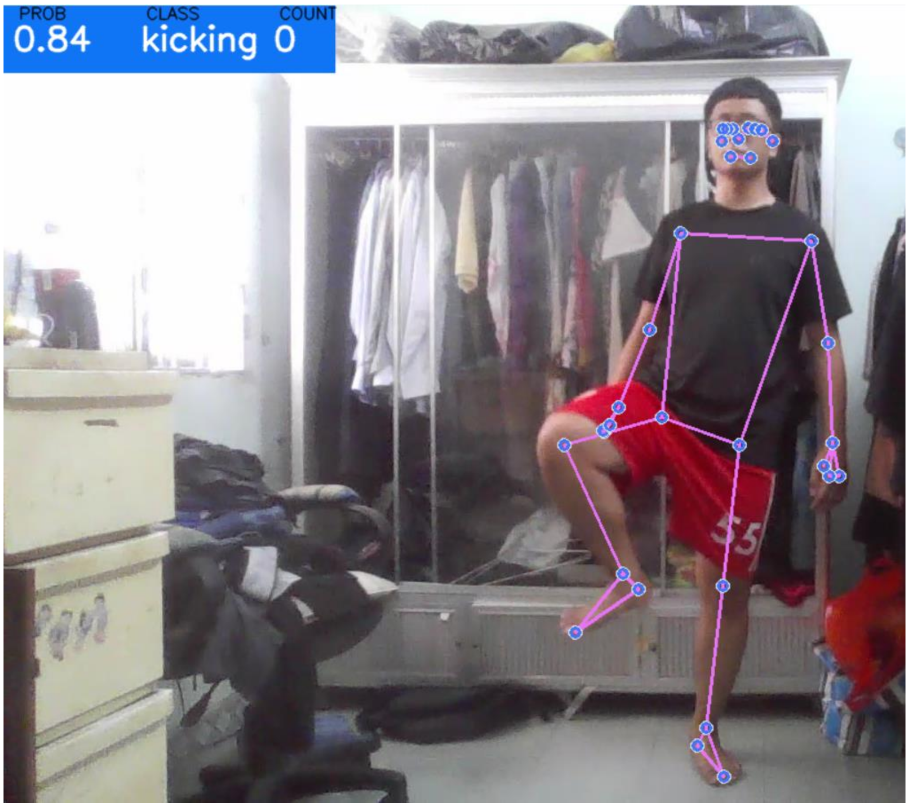


***Figure 32.*** *A person is performing a kicking action.*

**Choking**

Base on the metrics in **Table 10**, the best model for this type of action is Logistic Regression (LR) and the model give the following result

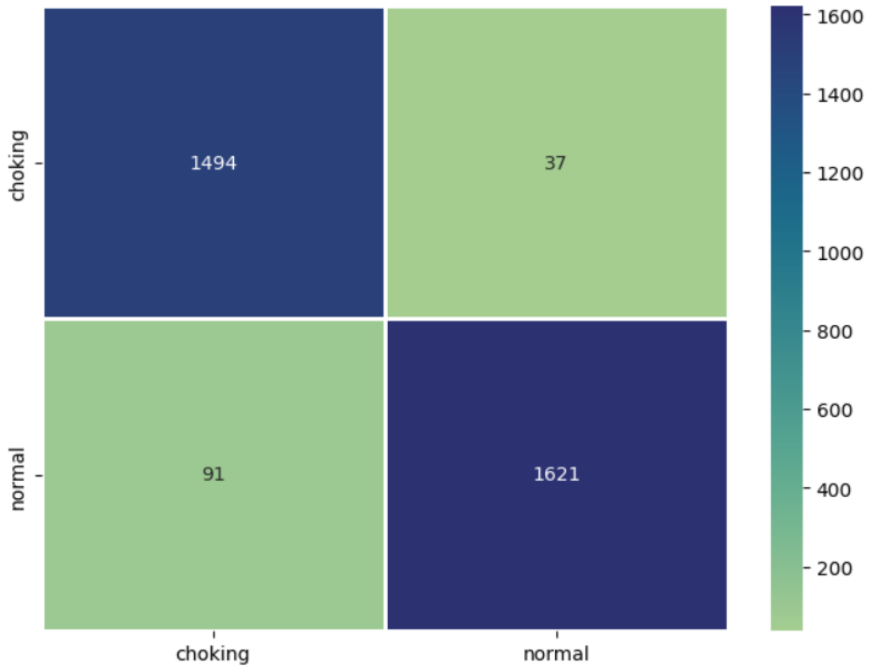


***Figure 33.*** *Confusion matrix of Choking action*

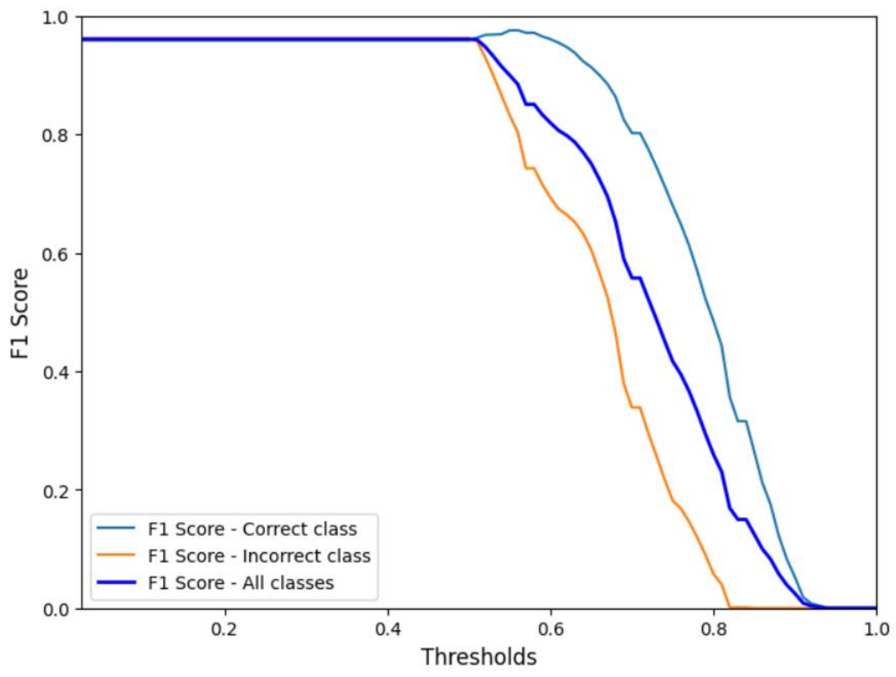


***Figure 34.*** *Choking action - F1 curve*

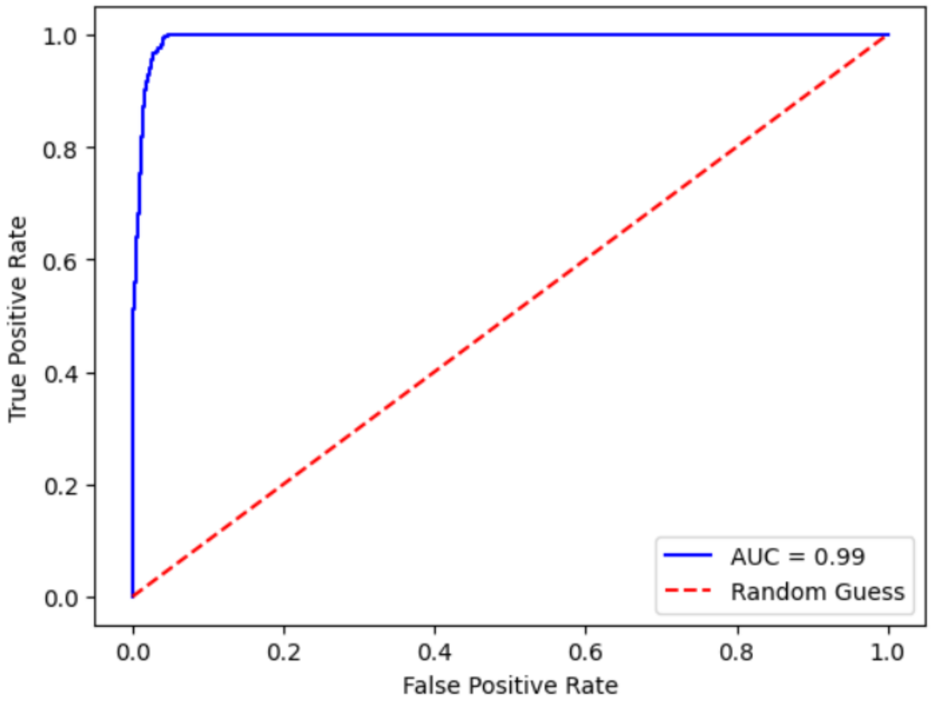


***Figure 35.*** *Choking action ROC curve*

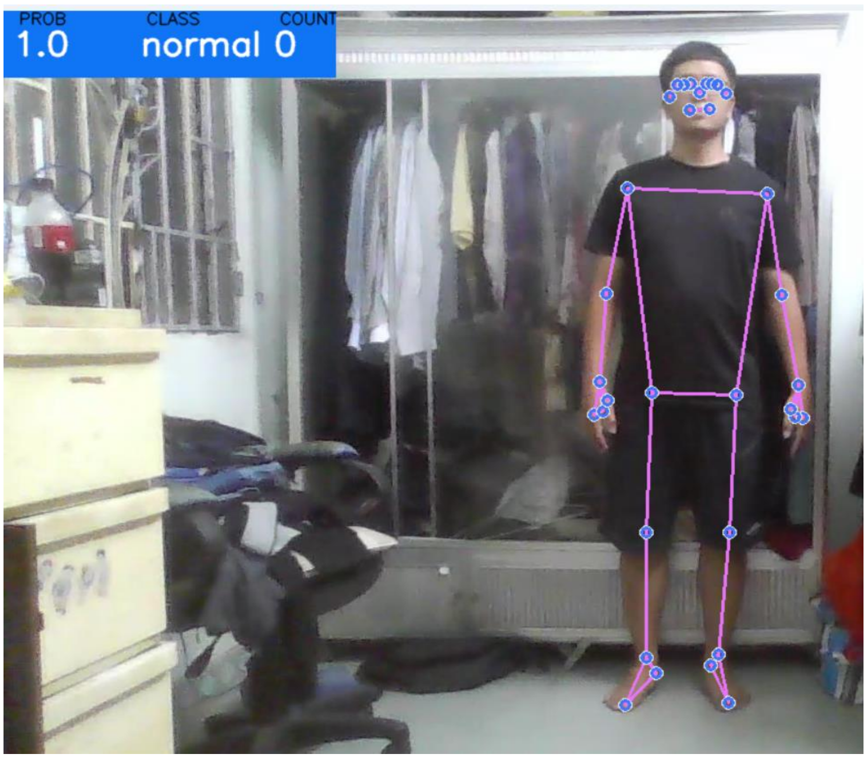


***Figure 36.*** *A man is standing normally*

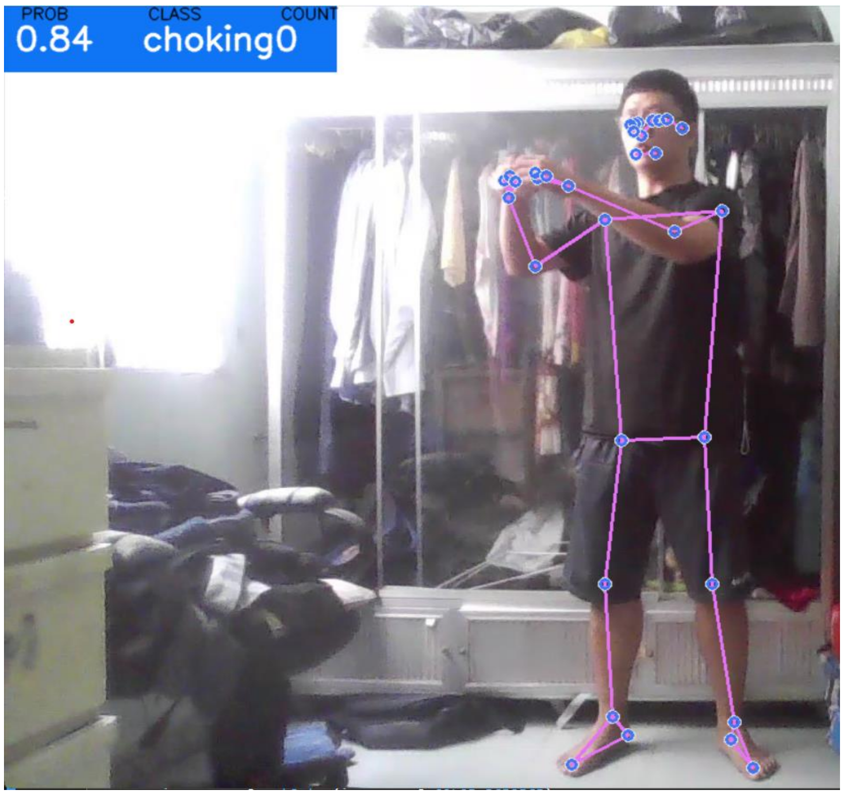


***Figure 37.*** *A man is performing a choking action*

# CHAPTER 4. CONCLUSION

## 4.1. FINAL RESULTS

After performing Building A Model For Detecting Violent Behavior In Children research, I was able to get accustomed to many machine learning theories, technology, and Python frameworks. The result that I possess is:

- Successfully building a model for detecting violent children in both realtime and offline.
- Achieve all objectives listed in **Chapter 1**.
- Selected the optimal models to perform this research.

The optimal models among 3 YOLOv5 models, 3 YOLO8 models and 3 distinguish violence detection model using Mediapipe and Scikit-learn are YOLO8m due to its high accuracy, F1 score along with minimal mistakes in detecting violence action shown in Confusion Matrix.

## 4.2. FUTURE WORKS

Due to limited time and resources, I have mainly focused on developing and finalizing the basic movement in every action. On the other hand, I am also preparing some future ways to improve the system in the future:

❖ Extending the datasets to include other participants' videos to eliminate the biases of the models and increase the accuracy.
❖ Adding model to detect more people in frames and calculate the distance between the person performing violence action to the children.
❖ Researching other violent actions that are performed in children.
❖ Connect with the alarming system or connect with parents' smartphones.

-

## REFERENCES

[1] Nadia Mumtaz, Naveed Ejaz, Shabana, "*Violence Detection Between Children and Caregivers Using Computer Vision",* Link: https://arxiv.org/pdf/2209.11680.pdf

[2] Maria Glenski, Ellyn Ayton, Josh Mendoza, Svitlana Volkova, "*Multimodal Violence Detection in Child-Adult Interactions*", Link: https://shorturl.at/vxzEN

[3] Thomas Davidson, Dana Warmsley, Michael Macy, Ingmar Weber, "*Automated Hate Speech Detection and the Problem of Offensive Language*", Link: https://arxiv.org/abs/1703.04009

[4] Tommaso Fornaciari, Luca Luceri, Emilio Ferrara, Dirk Hovy, "*Leveraging Social Interactions to Detect Misinformation on Social Media*", Link: https://arxiv.org/abs/2304.02983

[5] Liang Ye, Tong Liu, Tian Han, Hany Ferdinando, Tapio Seppänen, and Esko Alasaarela, "*Campus Violence Detection Based on Artificial Intelligent Interpretation of Surveillance Video Sequences*", Link: https://shorturl.at/qHMQ1

[6] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do & Kaori Togashi , "*Convolutional neural networks: an overview and application in radiology*", Link: https://shorturl.at/hLOS5

[7] "*Developing AI apps with Convolutional Neural Networks and OpenCV*", Link: https://shorturl.at/dhjq1

[8] "*CNN | Introduction to Pooling Layer*", Link: https://shorturl.at/djEO9

[9] "*Choosing number of Hidden Layers and number of hidden neurons in Neural Networks*", Link: https://shorturl.at/atJZ5

[10] "*YOLO : You Only Look Once – Real Time Object Detection*", Link: https://shorturl.at/kqxV8

[11] "*Do Thuan EVOLUTION OF YOLO ALGORITHM AND YOLOV5: THE STATE-OF-THE-ART OBJECT DETECTION ALGORITHM EVOLUTION OF YOLO ALGORITHM AND YOLOV5: THE STATE-OF-THE-ART OBJECT DETECTION ALGORITHM*", Link: https://shorturl.at/pswSW

[12] "*PANet: Path Aggregation Network In YOLOv4*", Link: https://shorturl.at/eyRX4

[13] "*Mosaic*", Link: https://shorturl.at/bkQ36

[14] "*MediaPipe is an Open Source Perception Pipeline Framework Developed by Google*", Link: https://shorturl.at/cfD14

[15] "*Person Pose Landmarks Detection Using Mediapipe*", Link: Person Pose Landmarks detection using Mediapipe - ML Hive

[16] "*On-device, Real-time Body Pose Tracking with MediaPipe BlazePose*", Link: https://shorturl.at/acgqw

[17] "*Human Pose Estimation Using MediaPipe Pose and Optimization Method Based on a Humanoid Model*", Link: https://www.mdpi.com/2076-3417/13/4/2700

[18] "*HOT DEBATE Should you spank your child? This is how physical punishment 'could harm your kids – for up to 10 YEARS*", Link: http://surl.li/mpwgi