Human Action and Face Recognition Using OpenCV Based on IoT

For Safety and Management School System

Prof. Mohab Mangoud and Asma Shayea
Department of Electrical Engineering, University of Bahrain, Kingdom of Bahrain
Corresponding author: Asma Shayea (e-mail: 202201062@stu.uob.edu.bh)

Abstract
In this paper, we proposed a system for the school environment that uses Open Source Computer Vision Library (OpenCV) to ensure high safety and management. The system suggests two solutions firstly, a school attendance system using a face recognition technique for automatically recording attendance and sending the record to the database using Message Queuing Telemetry Transport (MQTT). We also present a human recognition system to recognize abnormal or dangerous behaviour in school focusing on fighting action and for that, we used video classification based on Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN).

Keywords
Human Action Recognition, Video Classification, Haar Cascade Classifier, Convolution Neural Network, Recurrent Neural Network.

1 INTRODUCTION

These days, safety and security have the highest priority in different areas of our life such as home safety, school, data and information, personal privacy, and transportation. School is the place that joins our future youth with different goals and ambitions for the future. The technological revolution can help to increase the level of security. With the rapidly growing of Internet of Things (IoT) and the increase of sensor applications, sensors are expected to be embedded everywhere around us, resulting in a massive output of continuous and real-time data. Getting and sharing different types of data has been easier using low-cost internet connectivity and low-power devices. Computer vision with IoT can create a high-security system by utilizing camera devices as a sensors to detect, recognize or collect the required data and actions from human behaviour or any other object in the world and then send these data using the internet to be saved for analysing or sending alarms to other devices in IoT.

2 MOTIVATION

In recent decades, many unfortunate accidents have happened at schools resulting in the death or harm of the students. School is supposed to be a child's second home. Parents want to be reassured when sending their children to an educational environment that is expected to be monitored under a high level of supervision to achieve the maximum degree of protection. Every student deserves an encouraging and safe environment to learn.

3 PROBLEM

Students incidents such as children being forgotten and locked in school buses, incidents of death after a physical fighting between students, or shooting incidents in schools can be prevented with intensive monitoring and continuous communication with parents. These incidents make us worry about the loss of safety and security in many schools. In 2019, a Youth Risk Behaviour Survey (YRBS) was performed by CDC's on high school students across the United States. [1] According to YRBS results, 25.5% of California high school students had been in a physical fight and 12.3% of students were threatened or injured with a weapon on school property.

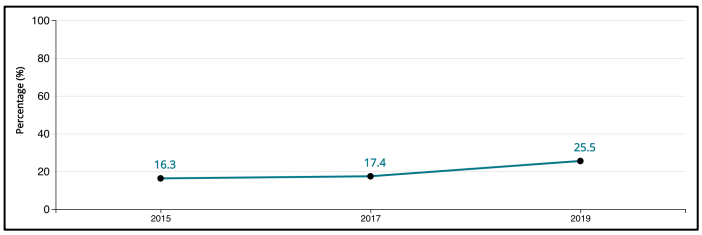

Fig. 1: High school students who were in a physical fight by YRBS results [1]

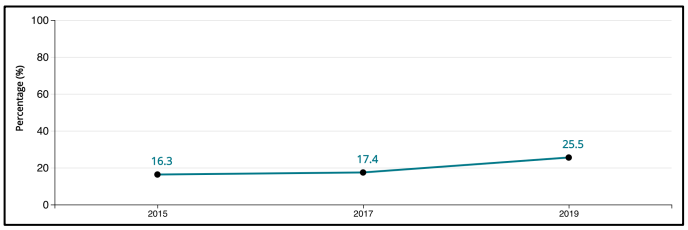

Fig. 2: Students who were threatened or Injured with a weapon by YRBS[1]

4 TOOLS, METHODS, AND ALGORITHMS

For real-time detection, we used a built-in 720p webcam camera in a MacBook Pro device as an IoT sensor.

4.1 MQTT (Message Queuing Telemetry Transport)

MQTT is a standard publish/subscribe messages protocol of IoT used to communicate between devices with limited network bandwidth.[2] Some of the basic concepts of MQTT include:

1- **Publish/subscribe**: Also known as pub/sub. Publishing is the process to send messages while subscription means receiving those messages by MQTT clients on a topic that it subscribed to and related to that topic.

2- **Topic and subscription**: When the publisher sends a message on a specific topic, all subscribers to that topic will get that message.

3- **Quality of service levels (QoS)**: QoS is the level of the quality of service within two message parties considering the assurance of data distribution. The system need will determine which level needs to be used.

4- **Broker**: The broker is responsible for controlling the transformation of information by receiving all messages from the publisher, filtering them and then sending them to all interested subscribed clients.

4.2 Mosquito Broker

The broker is the link that connects between devices and the system. The subscription handles sessions, missed messages, and security including authentication and authorization. There are many different brokers for different purposes. In our system, we used Mosquito Broker which is an open source that implements MQTT protocol and is suitable for all lower power computers to full servers.

4.3 Haar Cascade Algorithm

Haar cascade is a popular algorithm for facial detection and recognition. It can detect one or more faces at the same time. It is a method that use Haar features inputs into a series of classifier (cascades) to identify faces. This method identifies only one type (e.g. Edge or lines) but several of them are used in parallel to detect labels like eyes and face together. [3] There are four phases:

1- **Selecting Haar-like features**: A window will slide over the whole image to apply the Haar-like feature on each of image window to extract the features for the human face. It partitions the face into regions considering the size and brightness variation. Each feature results in a value that is calculated by subtracting the addition of the white zone from the addition of the black zone.

Value = **Σ**(pixels in black zone) - **Σ**(pixels in white zone)

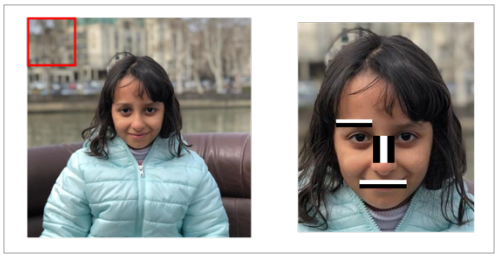

Fig. 3: Applying Haar-like features on an image

2- **Create an integral image from the original image**: For the large images, instead of calculating submission for each pixel in the rectangle, we only have to calculate the four edges of the rectangle.

3- **AdaBost**: Reduce the number of features that we do not need in order to identify the face. It works

by training some images with faces and others without faces to classify them, thus getting the desired information.

4- **Cascade Classifier**: It gets rid of non-face candidates and filters them by passing all images' sub-window into different phases. The candidate that passes all phases, will be detected as a face.

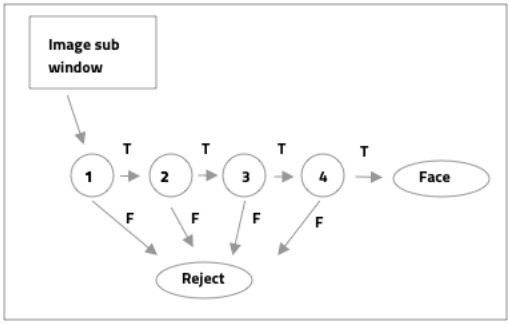

Fig. 4: Cascade Classifier

4.4 Convolutional Neural Network (CNN)

CNN is an artificial neural network using popularity for image recognition and classification. It consists of four types of the following layers:

1- **Input layer**: Input layers contain image data. It holds the pixel value of the image.

2- **Convolution layer**: Images parts (a few pixels at a time) will pass to this layer to apply some filters in order to extract features(e.g. edge, object…etc) and perform a convolution operation calculating the dot product of the original pixel values for the input image with weights defined in the filter. The output will be fed to the next layer as an input.

3- **Pooling layer**: This layer reduces the spatial volume for the inputs that come from the convolutional layer output. The filter will again pass over the results from the previous layer by applying Max Pooing and selecting one value from each of the groups of values (typically maximum). As the convolution layer generates a matrix that is smaller than the original image, this layer will further reduce the size of that matrix. This will make training much faster by focusing on the most important features.

4- **Fully-connected layers**: Classifies images into categories by training using the output from previous layers as an input. The output will be labelled with different probabilities and the height probability will be the classified label.

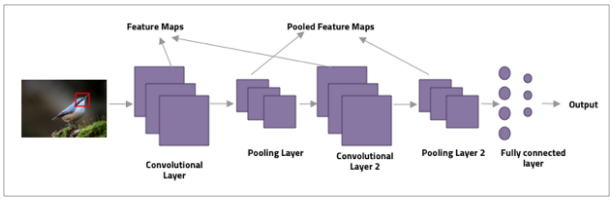

Fig. 5: CNN

4.5 Recurrent Neural Network (RNN)

RNN is a type of neural network used to model sequence data. Traditionally in neural networks, all input and output are independent, however, in RNN with the hidden state features, it can remember some information for sequences. In other neural networks, each hidden layer has its own weight and biases whereas in RNN each hidden layer has the same weight and biases, which reduces the increase of parameters. Since weight and biases are the same for hidden layers that means each hidden layer has the same characteristic. Rather than create many hidden layers, it will create just one and loop over it as many times as needed.

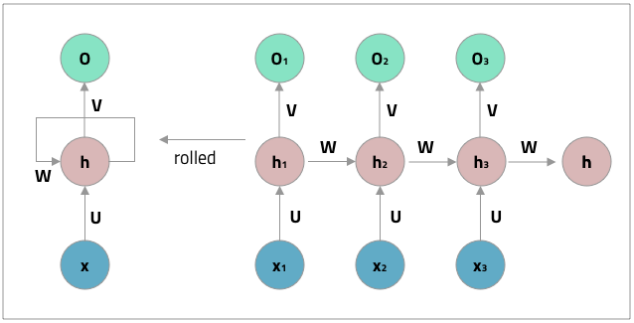

Fig. 6: RNN

5 PROPOSED SYSTEM

The proposed system in this paper is a system for the school environment. This system uses open-source computer vision (OpenCV) based on IoT to provide two solutions that would contribute to preventing some of the primary safety-related incidents that can happen at schools. In addition, these solutions will be useful to facilitate work and reduce costs by automating some tasks. The following are two proposed solutions for the safety and management school system.

5.1 Face Recognition Attendance System

The workflow of this system is clarified in Fig.7. A camera is placed in the classrooms to record attendance. At 6 a.m., the camera will start detecting one or multiple faces in the room and take attendance by capturing and recognizing those faces it has in its database. Once it has recognized any recorded student face, it will automatically mark the student as in attendance. At 9 a.m., attendance records will be sent to the server to save. At this time, alarm messages will be sent to any absent/unrecognized students' parents to inform them.

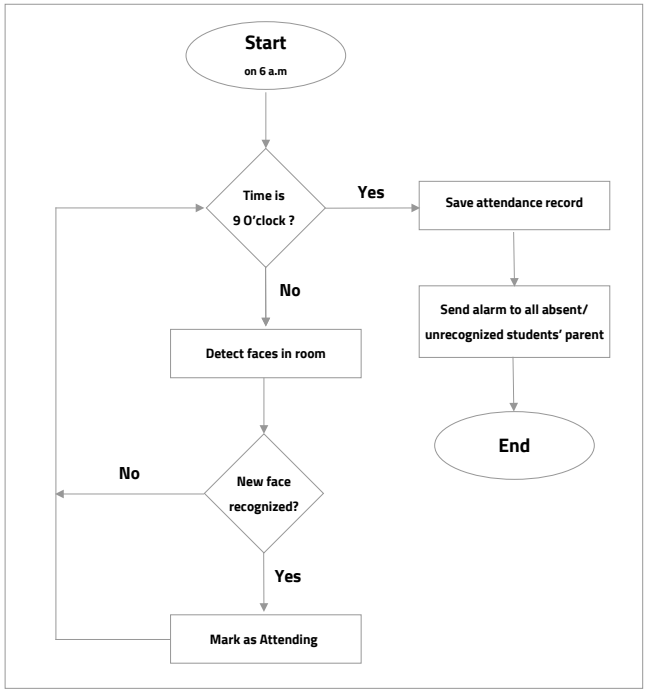

Fig. 7: Face Recognition Attendance System

5.1.1 Implementation and Results

By using Haar Cascade Algorithm as explained in section 4.3, we can recognize multiple faces in a room using a real-time webcam. Results are shown in the following images:

1- Faces detected and recognized by real-time webcam.

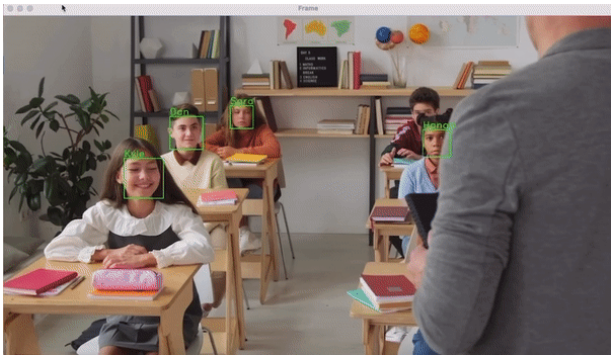

Fig. 9: Face recognition tested on a video for students in a classroom

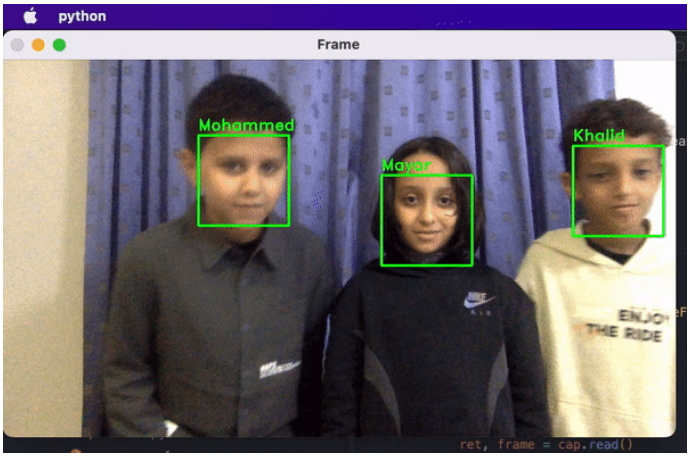

Fig. 8: Faces recognition tested by webcam

2- Each recognized student will be marked as attending.

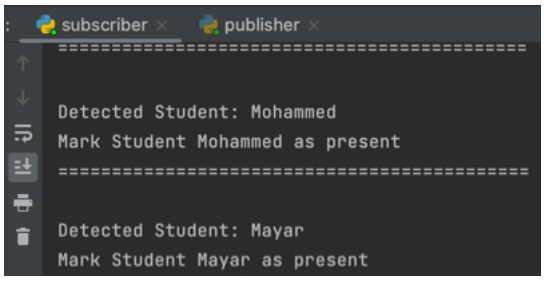

Fig. 10: Mark recognized the student as present

3- At 9 a.m., attendance data will be sent to the system by MQTT for updating.

	A	B	C
	Student_name	**Attendance**	**Phone**
	Sami	Absent	598233830
	Asma	Absent	5362000076
	Mohammed	Present	5746333825
	Khalid	Present	5664646473
	Mayar	Present	5554320076
	Joory	Absent	5636646473

Fig. 11: Record updated

4- For all absent/unrecognized students, their parents will receive a notification.

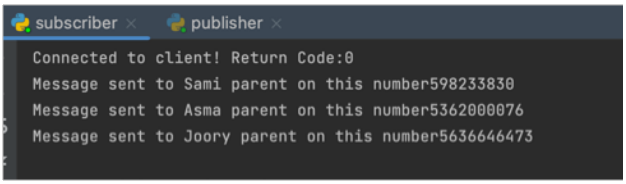

Fig. 12: Notification sent

5.2 Fighting Action Recognition System

With human action recognition, we used video classification to detect any abnormal or dangerous behaviour from students. In this paper, we will recognize fighting as the human action. Using the real-time camera, once the system recognizes any current fighting action happening at the school, it will send an alarm to the security office.

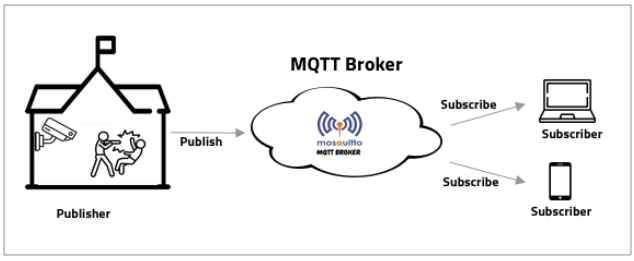

Fig. 13: Fight Action Recognition System

5.2.1 Dataset

The dataset for this paper was collected from the Kaggle for the fighting action. It includes two labels. One folder contains 100 fighting action videos and the other contains 100 videos for non-fighting action.

5.2.2 Human Action Recognition Implementation using video classification

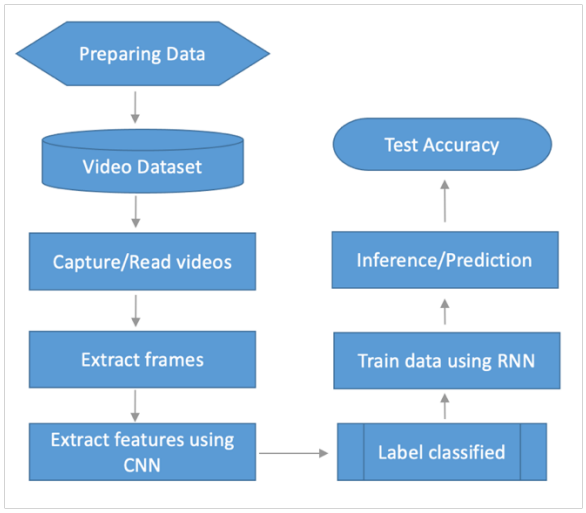

Fig. 14: Implementation Flowchart

Using video classification with CNN and RNN algorithms as explained in sections 4.4 and 4.5, we were able to achieve good results by recognizing different fighting actions.

Video is a collection of frames in a specific order. There are many approaches for applying video classification. The approach we applied here is CNN-RNN. For RNN, we used a type of RNN called Long Short Term Memory (LSTM). It can remember the context for long input sequences. Following is the implementation of human fighting action recognition.

1- **Preparing Data**: This is done by splitting the dataset into a training set with 80 videos with fights and 80 videos without fight action, and a validation set with 20 videos with fights and 20 videos without fight action.

2- **Prepare and Read videos to extract frames**: After extracting frames, we set our hyperparameters as follows:

3-

- Maximum frame count to be **20**. As the number of frames will differ in each video which would prevent stacking them into batches, we set a fixed number of 20 frames. Therefore, if any video has a lesser number of frames, we will pad the video with zeros.

- Image size. Image size must be fixed for each frame because the neural network needs them at the same time for the CNN feature extractor.

- The number of features. The number of extracted features from each video will be 2,048.

Therefore, if we have **200** videos, the total number of train data will be **20*2048*200**.

4- **Extract features**: With the CNN algorithm, we passed the frames images to CNN layers to extract image features thus the output will be a label for each frame as either fights or noFights as shown in Fig. 14.

Fig. 15: Label Classified

5- **Feed data to LSTM sequence model**:

We will use the output of the independent CNN extracted information and feed them to the LSTM layer for training which will fuse them temporarily. LSTM has the ability to identify temporal relations between these frames.

6- **Validation**: After training, we evaluated these results with our test data to check the test accuracy.

5.2.3 Training Results

As shown in Fig. 15 and Fig.16, the accuracy increased to 95% and the loss decreased to 21.04% after 30 Epoch.

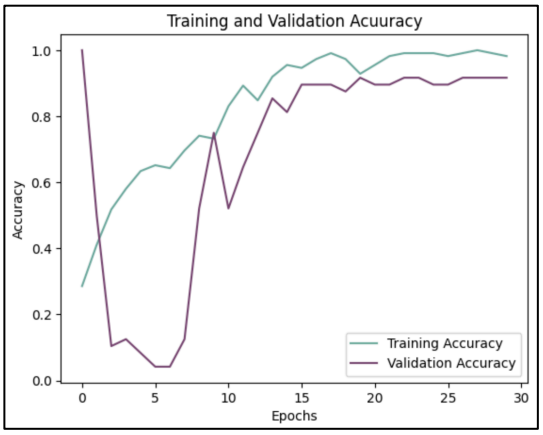

Fig. 16: Accuracy in training and validation

Fig. 17: Loss in training and validation

5.2.4 Test Results

For testing, we randomly selected some videos as a sample from the test dataset to predict and recognize the fight action. Following are some results for the different videos selected.

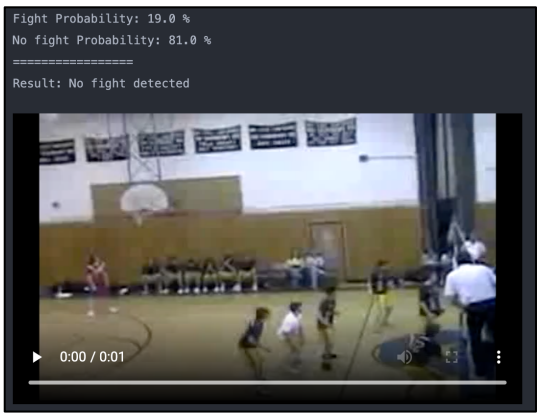

Fig. 18: Result on a non-fighting video – 81% not a fight

Fig. 19: Result on a fighting video – 96% is a fight

Fig. 20: Result on a non-fighting video – 80% not a fight

Fig. 21: Result on a fighting video – 84% is a fight

6 CONCLUSION AND FUTURE WORK

This work reviewed the most widely OpenCV algorithms including Haar Cascade for face recognition, CNN, and RNN for video classification to recognize human actions. We are looking to improve these techniques to produce more accurate results. We also hope, with this technical revolution, that we can provide a safer environment for the student in schools and other different areas. We are looking forward to further developing this system by adding more actions and objects (e.g. weapons) to be recognized and detected, in addition to other atypical human behaviours (e.g. smoking) to prevent any potential risk can happen to our youth.

7 REFERENCES

[1] Explore Youth Risk Behavior Survey Questions - United States, 2019 (YRBS)

[2] Soni, Dipa & Makwana, Ashwin. (2017). A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT).

[3] Aydin, Ilhan & Othman, Nashwan. (2017). A new IoT combined face detection of people by using computer vision for security application. 1-6. 10.1109/IDAP.2017.8090171.

[4] Karpathy, Andrej & Toderici, George & Shetty, Sanketh & Leung, Thomas & Sukthankar, Rahul & Fei-Fei, Li. (2014). Large-Scale Video Classification with Convolutional Neural Networks. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 1725-1732. 10.1109/CVPR.2014.223.

[5] Wang, Jiang & Yang, Yi & Mao, Junhua & Huang, Zhiheng & Huang, Chang & Xu, Wei. (2016). CNN-RNN: A Unified Framework for Multi-label Image Classification. 2285-2294. 10.1109/CVPR.2016.251.

[6] O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.

[7] Shaik, Jahanara. (2021). Detecting autism from facial image. 10.13140/RG.2.2.35268.35202.

[8] Fan, Yin & Lu, Xiangju & Li, Dian & Liu, Yuanliu. (2016). Video-based emotion recognition using CNN-RNN and C3D hybrid networks. 10.1145/2993148.2997632.

[9] Ganesan, Srividhya & Dr, Raju & J, Dr. (2021). Prediction of Autism Spectrum Disorder by Facial Recognition Using Machine Learning. Webology. 18. 406-417. 10.14704/WEB/V18SI02/WEB18291.

[10] Video Fight Detection Dataset | Kaggle