Faculty Of Engineering

Electronics, Communication, and Computers Department

(Computers Section)

Helwan University

SMART SURVIELLANE SYSTEM FOR CRIMINAL BEHAVIOR (S^3CB)

Presented By:

-Abdelrahman Yasser Ahmed.

-Abdallah Mahmoud Ahmed.

-Ahmed Talat Ibrahim Faid.

-Abdallah Mostafa Mohamed Elkilany.

-Omar Raafat Mostafa Sayed.

SUPERVISED BY:

DR. Maher Mansour.

2022

Individual Contributions

Name	Contributions
Abdelrahman Yasser	-Building and training CNN model for feature extraction. -Building RNN for final classification layers. -Building CDI and information entropy. -Test YoloV3 for weapon and fire detection and compare it with the proposed model. -contributing to Dataset Annotation and Data augmentation. -spend a great time in the anomaly detection researches. - contributing to Building Database Design for the whole system. - contributing to Building important Diagrams for the whole system such as use case diagram and others. -Organize and review all the Documentation of the system.
Abdallah Mahmoud	-Reviewing and Classifying Dataset. -Dataset annotation. -System Documentation. -Deploying DL Model on a Raspberry-Pi Chip.
Ahmed Talat	-Create real time database and telegram bot to let ESP interact with the system - write full code for ESP - connect actuators and design the maquette manually - participate in downloading datasets and in research - write the action part in the book - helped in website Application
Abdallah Mostafa	-Building the front UI of the web and mobile applications for client and admin. -Building the back end of the web and mobile applications for client and admin. -Building the database, and API that manipulates everything about communication in the system. -Video streaming using multiple IP cameras to a web server. -Deployment of the system over a local network and allowing access from outside the network using port forwarding. -Building use case diagram and sequences diagrams and ER diagram.
Omar Raafat	- Participating in Dataset Annotation. - Pre-processing the dataset such as augmentation. -Building and training CNN model (MobilenetV2) for feature extraction. -Building RNN(LSTM) for final classification of Anomaly events such as Shoplifting, Explosion, Fighting. -Real-time recognition using multiple IP cameras. -Send result of real-time using REST API. -deploying the Model to web using Flask. -Write Project presentation.

Abstract

Many surveillance cameras are almost spread everywhere for security purposes, but the crimes took place in a very short time. The big problem that the crime is not detected at the same time it Occurs, but after analyzing the recording videos of the surveillance cameras that takes a lot of times to discover what happened so the security guards could not discover and respond to the crime instantaneously. Moreover, the attention of the Security guards will be degraded with time which opens the chance for the criminals to start their mission easily and successfully.

Our proposed system is an entire system that could recognize the anomaly behavior instantaneously by sending a captured screenshot image for the anomaly behavior, the time and date, the anomaly type, and the zone ID. By these notifications the guard can take a proper action also automated actions will be taken based on the anomaly type. Also, our system has a control room where the admin can add security agents, remove security agents, edit security agents' details such as changing agent from one zone to another zone, also providing live streaming to watch out all the surveillance cameras in the system.

So, our proposed system will be a smart surveillance system for criminal behaviors that plays an important role in detecting and recognizing anomaly behaviors quickly, efficiently, notifying the security guards, and taking automatic actions.

Table of Contents

List of Tables

List of figures

Introduction

Smart systems now change our life and make it easier. In our proposed system we are focusing on smart security systems as it is especially important to use this technology to save human lives. As the number of cameras increases day by day, not corresponding to the number of the security agents [1][2] and with the percentage of the human error as it is impossible for the security agents to be in the same attention level all the time.

From that point it is necessary for smart systems to be used to solve the mentioned problems. As many organizations and companies have used real-time surveillance with machine and deep leaning fields, and much research has also proved that anomaly recognition can achieve a very satisfactory result, the existing intelligent systems still generally faces many challenges properties of real-time recognition:

- Efficiency: as for real time applications carrying about how to reduce the time of processing one frame which can be called frame rate, and the required number of frames make an accurate prediction, and the training time to decrease it so system will be easy to update it.

- Generalization: as it is so difficult for any benchmark dataset to cover all the anomaly types as in the current benchmark dataset used like UCF-Crime[3] , UCSD [4], UMN [5], CUHK [7], and SUBWAY [6] , 2018 AI City Challenge[9] all these dataset cannot cover all the anomaly all over the world the challenge is how to achieve the generalization to the types of the different types of the anomaly and that's are the point to find the similarities between different classes which could reduce the number of cases the model has to learn, while making sure its generalization ability property has been achieved.

- Update during the runtime: Only a few real-time detection algorithms have adopted online update. Due to the rapid changes in society, the false alarmed videos in the real monitoring system should be fed back to the training process to continuously improve the accuracy and generalization ability of the system [8].

The major contributions of our proposed System can be summarized as follows:

- Selecting a proper model between the widely used deep learning models as in the literature review Taking in the consideration the speed and the time complexity while training and while prediction [efficiency and accuracy].

- Modified dataset: We filtered the dataset from many problems then after that we annotated our dataset to extract the frames that are related to the pattern of the anomaly and collect similar anomalies together in the same class.

- Building a fully functional Control Room With admin page that can add clients (Security agents) or removing clients or editing client details and live monitoring.

- Building a mobile application which will represent the security agent so any crime that occurs will be sent to the client agent directly.

- Building the database for the entire system.

- Building hardware actuators that are implemented based on each type of anomaly.

1. Literature Review

Detecting anomalies according to different machine learning models and abnormal situations have proved themselves in that field in recent years [10], [11],[12]. Deep learning models such as LSTM [13], [14], [15], and a lot of papers used a combination from CNN and LSTM Models which have proven its progress in anomaly detection and recognition as mentioned in many researches as they focused using different CNN models comparing between them in terms of efficiency and accuracy for extracting spatial features from the image after that extracting temporal features using different RNN models comparing between them in terms of complexity and the accuracy for detecting the sequence of the frames correctly as it is represented as the final classification network for example the framework has been implemented as follow according [21] framework is divided into two parts the first part is feature extraction using a pre-trained ResNet-50, the second part its input is the output of the first part receiving feature vector by a sequence model [RNN using BD-LSTM] as in fig 2 and that framework used for anomaly detection where they gather all the 12 cases in one class called anomaly class and one class for the normal case, so the final layer is a binary cell used for the final prediction(classification).

Another research paper [22] used the first part inceptionV3 as the CNN model for spatial feature extraction and the second network is LSTM model for temporal feature extraction, using these frameworks to recognize between 13 different anomaly class each anomaly in the 13 class considers as a certain unique class with the normal case as in fig 1. And another research paper [23] used the first part mobileNetV2 s the CNN model for spatial feature extraction and the second part for the network is Attention Residual LSTM for temporal feature extraction, that framework used for recognizing 5 different anomaly classes.

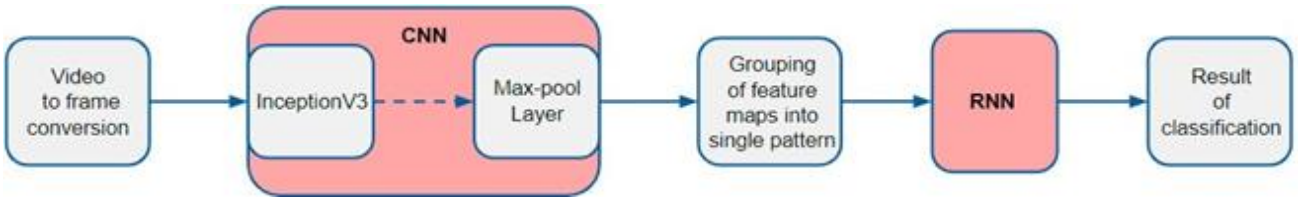

Figure 1 InceptinV3 feature extraction and RNN network (LSTM layer) Temporal feature extraction

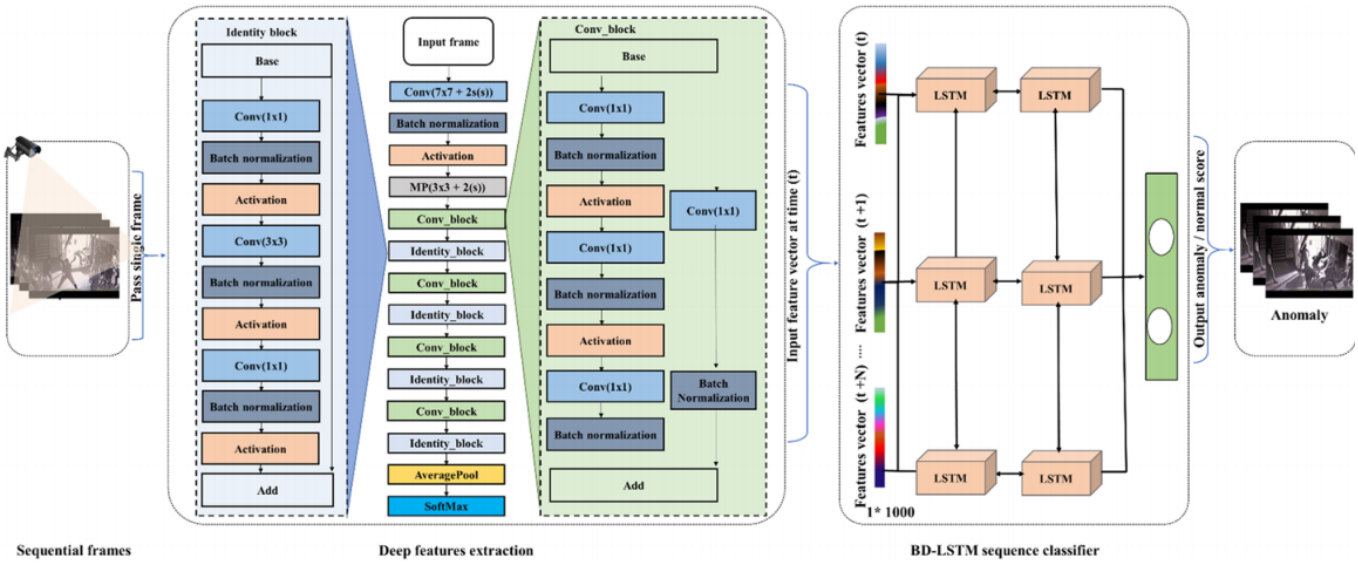

Figure 2 ResNet-50 feature extraction and RNN network (BD-LSTM layer) Temporal feature extraction

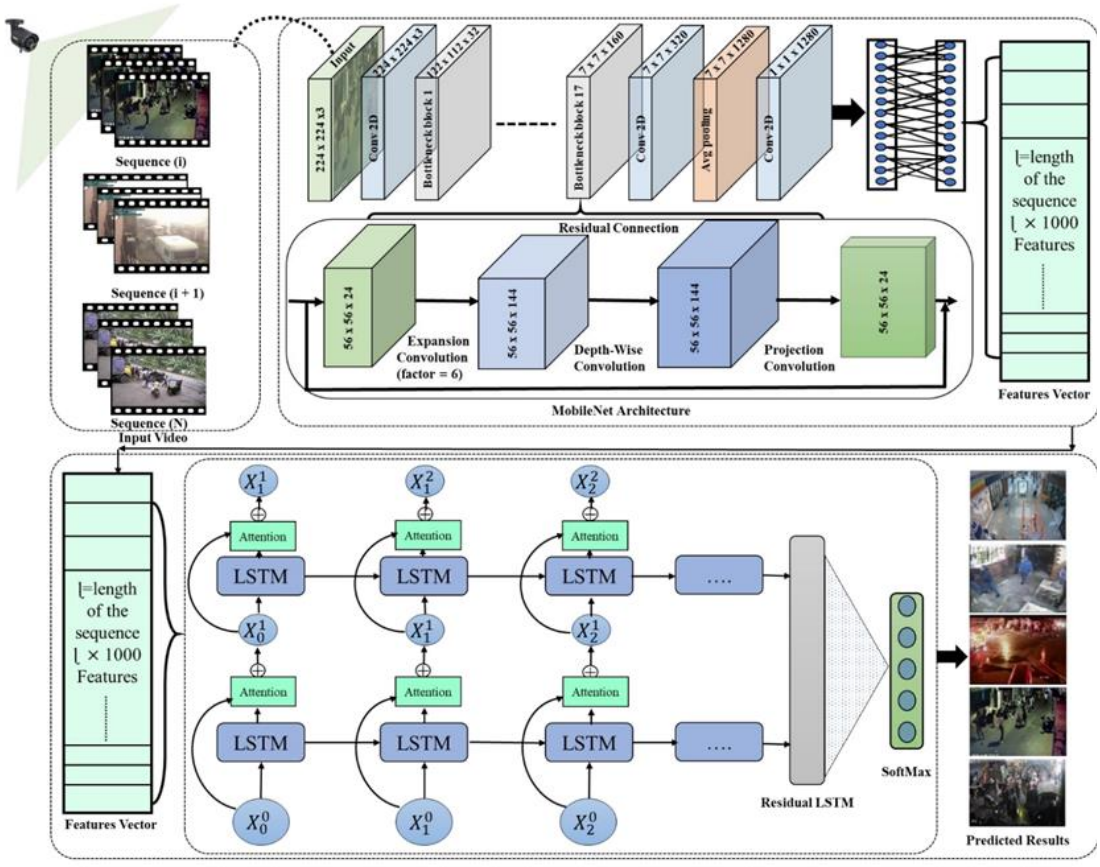

Figure3MobileNetV2 feature extraction, RNN network (attention-Residual LSTM) Temporal feature extraction

We have summarized [21][22][23] in an organized table like in Table 1 to compare different architectures using different metrics.

Model	Recall (%)	Precision (%)	ACC (%)	AUC (%)	F1 Score (%)	Time Complexity (Seconds)	#Param (10^6)	Model Size (MB)
Mobile Net V2 +LSTM [22]	86	74	-	88	77	-	-	-
Mobile Net V2 +BD-LSTM [22]	79	84	-	87	76	-	-	-
Mobile Net V2 + residual LSTM [22]	91	78	-	95	82	-	-	-
Mobile Net V2 +attention-residual LSTM [22]	78	87	78.43	96	81	0.263	3.3	12.8
InceptionV3+multiLayer LSTM [21]	-	-	97.23	-	-	-	-	-
VGG –19+RNN [23]	-	-	78	-	-	-	-	-
VGG –19+LSTM [23]	-	-	80	-		-	-	-
VGG –19+RNN + Multi-layer BD-LSTM [23]	-	-	82	-	-	0.22	143	605.5
Inception V3 + RNN [23]	-	-	71	-	-	-	-	-
Inception V3 + LSTM [23]	-	-	79	-	-	-	-	-
InceptionV3-BD-Multi Layer [23]	-	-	80	-	-	-	23	148.5
Res-Net50 + RNN [23]	-	-	78	-	-	-	-	-
Res-Net50 +LSTM [23]	-	-	84	-	-	-	-	-
Res-Net50 +BD-Multi Layer [23]	-	-	85.53	-	-	0.20	25	143

Table 1 Different Models Evaluation Metrics Review

Also, there are researchers goes to use Auto-Encoder [17], [18], [19] for anomaly detection, OCNN [20] and Markov Model [16] have been widely used for anomaly detection, Time-series has also been explored in anomaly detection like [24] and a research paper used anomaly detection and feature analysis based on time series [8] for anomaly detection as it focuses on how to achieve an Effective solution for the anomaly detection how the response to be as fast as possible for making a prediction also while training the algorithms that's has been done by the workflow as follows is shown as Fig. 4 First of all, the workflow contains two main stages feature extraction and feature description. The first stage is extracting the feature map from the frames, calculating the CDI (Combined Difference Image) [feature extraction based on CDI] as in fig 5 , the second stage is feature description based on information entropy where dimensionality reduction is a common process in to train on videos contain the anomaly or even normal videos as the high feature dimension contains redundant information and increases the training time on the two-class SVM or auto encoder as feature map of just

one image which is CDI[Combined Difference Image] is converted to a single value called frame energy which carry the energy of the CDI image [CDI is Described by that energy FE (Frame Energy)] then after that all the video will be described by the VEV (Video Energy Vector) and that vector will be entered on machine learning model such as SVM for training and then to predict whether the video has abnormal events or using Auto Encoder for training and classification.

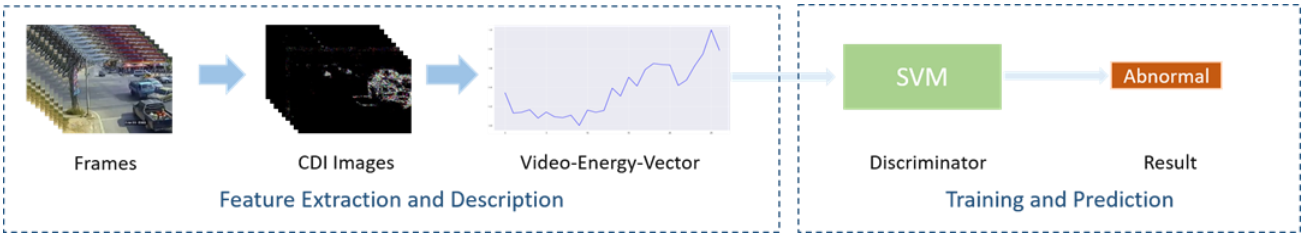

Figure 4 Feature Extraction and Description, training and prediction stages

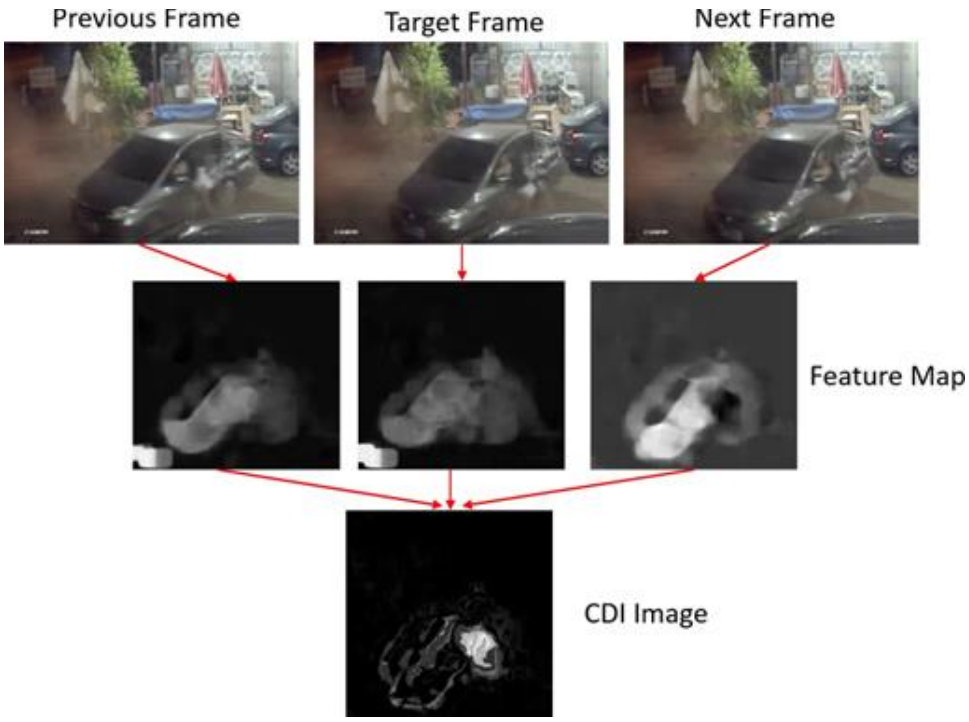

Figure 5 Extracting features and performing CDI Operation.

As shown from the previous works as mentioned in [8] most of the works fall into two categories: target-based and non-target-based where target based is to extract the objects from the frames and analyze them [25][26].

And the non-target based not focus on extracting the object, but instead training with processed fragments, images, or videos. [8][21][22][23] is non-target based. And our proposed solution also will focus on motion and the changes through the sequence of the frames.

Also, there are a lot of research focus on non-target based to just making object detection but not objects like pedestrian or cars which is under the anomaly types in some cases if found in certain zones, but another object detection is used under the non-target based is Weapon detection and fire detection which epically used for avoiding crimes to occur and a lot of research published on that topic like [27][28].

2. Module: Deep Learning

Machine learning and deep learning are the two main concepts of data science and the subsets of artificial intelligence. Most people think machine learning, deep learning, and as well as artificial intelligence as the same terminologies. But all these terms are different but related to each other.

In this topic, we will learn how machine learning is different from deep learning. But before learning the differences, let's first have a brief introduction of machine learning and deep learning

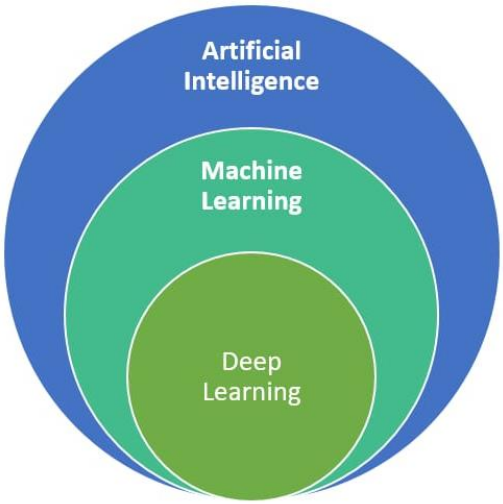

Figure 6 Artificial intelligence, machine learning and deep learning

2.1. What is Machine Learning?

Machine learning is a subset, an application of Artificial Intelligence (AI) that offers the ability to the system to learn and improve from experience without being programmed to that level. Machine Learning uses data to train and find accurate results. Machine learning focuses on the development of a computer program that accesses the data and uses it to learn from itself. [30]

The popular applications of ML are Email spam filtering, product recommendations, online fraud detection, etc.

There are various ways to classify machine learning problems.

- **Supervised Learning.**
- **Unsupervised Learning.**
- **Semi-supervised learning.**
- **Reinforcement learning.**

2.2. What is Deep learning:

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech, and audio, whereas recurrent nets have shone light on sequential data such as text and speech.[29]

2.3. Comparison of Machine Learning and Deep Learning

Parameter	**Machine Learning**	**Deep Learning**
Data Dependency	Although machine learning depends on the huge amount of data, it can work with a smaller amount of data.	Deep Learning algorithms highly depend on a large amount of data, so we need to feed a large amount of data for good performance.
Execution time	Machine learning algorithm takes less time to train the model than deep learning, but it takes a long-time duration to test the model.	Deep Learning takes a long execution time to train the model, but less time to test the model.
Hardware Dependencies	Since machine learning models do not need much amount of data, so they can work on low-end machines.	The deep learning model needs a huge amount of data to work efficiently, so they need GPU's and hence the high-end machine.
Feature Engineering	Machine learning models need a step of feature extraction by the expert, and then it proceeds further.	Deep learning is the enhanced version of machine learning, so it does not need to develop the feature extractor for each problem; instead, it tries to learn high-level features from the data on its own.
Problem-solving approach	To solve a given problem, the traditional ML model breaks the problem in sub-parts, and after solving each part, produces the result.	The problem-solving approach of a deep learning model is different from the traditional ML model, as it takes input for a given problem, and produce the end result. Hence it follows the end-to-end approach.
Interpretation of result	The interpretation of the result for a given problem is easy. As when we work with machine learning, we can interpret the result easily, it means why this result occur, what was the process.	The interpretation of the result for a given problem is very difficult. As when we work with the deep learning model, we may get a better result for a given problem than the machine learning model, but we cannot find why this particular outcome occurred, and the reasoning.
Type of data	Machine learning models mostly require data in a structured form.	Deep Learning models can work with structured and unstructured data both as they rely on the layers of the Artificial neural network.
Suitable for	Machine learning models are suitable for solving simple or bit-complex problems.	Deep learning models are suitable for solving complex problems.

Table 2 Machine Learning and Deep learning Comparison.

3. Module: Convolution Neural Network (CNN)

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs.

They have three main types of layers, which are:

- Convolutional layer.
- Pooling layer.
- Fully-connected (FC) layer.

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.[35]

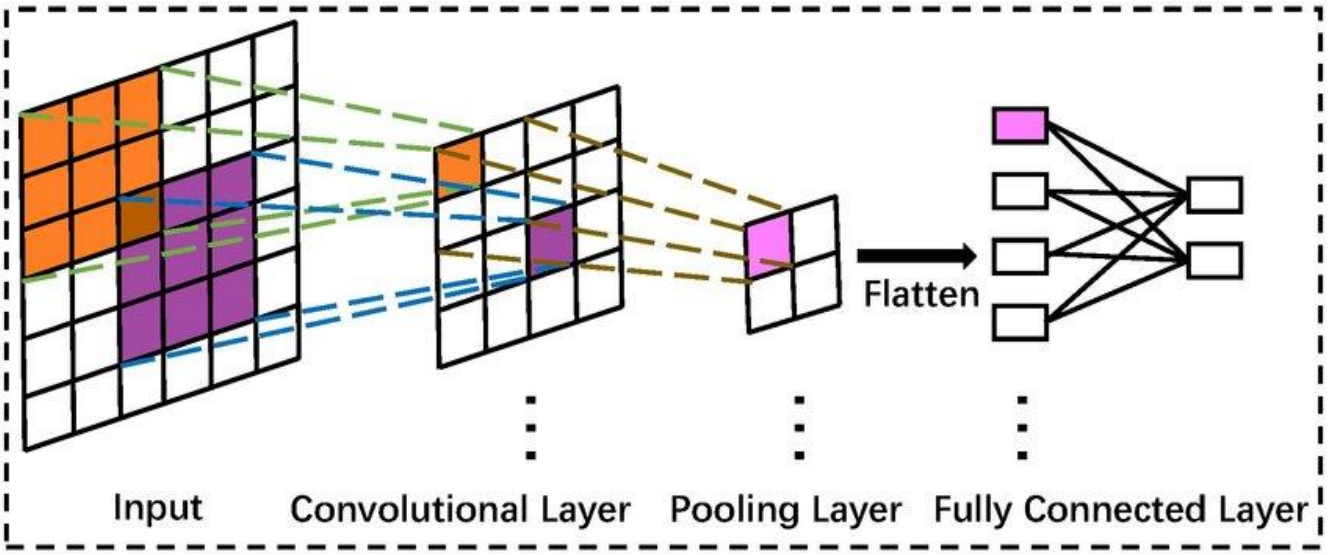

Figure 7convolution neural network different types of layers.

3.1. Convolutional Layer

The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let's assume that the input will be a color image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions—height, width, and depth—which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as convolution.

The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.[35]

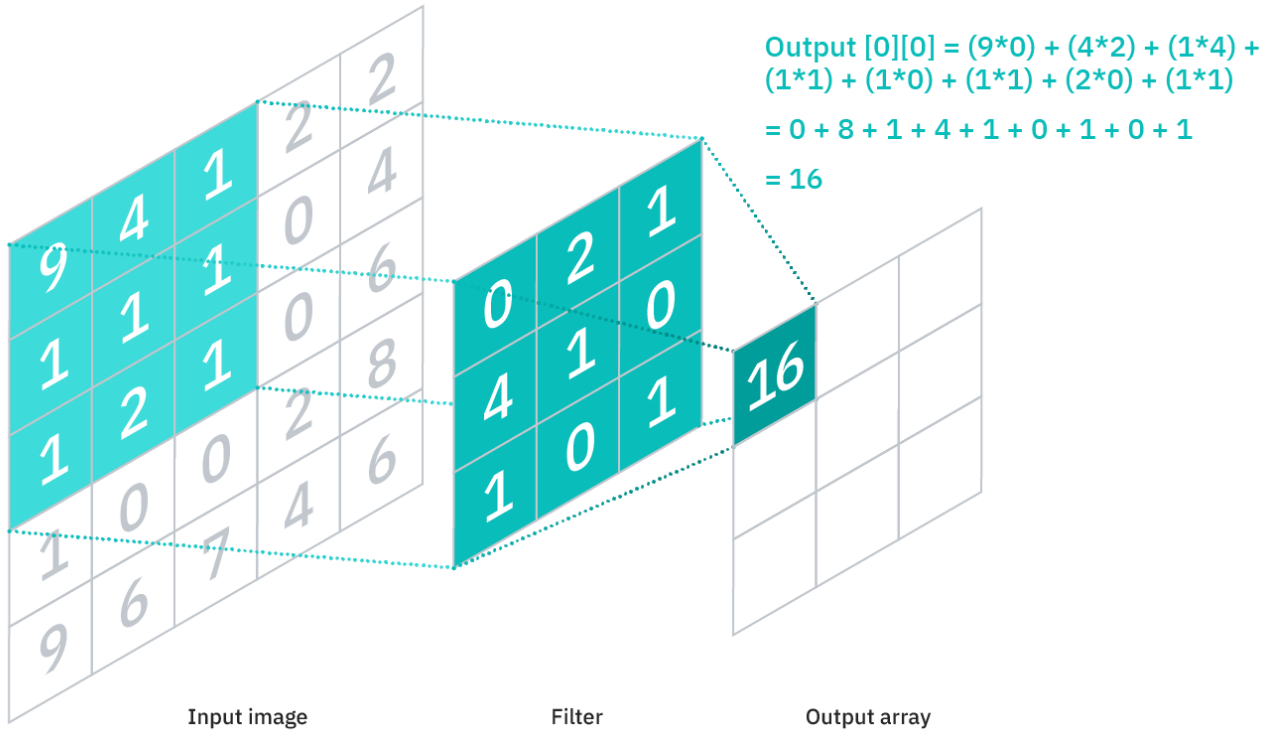

Figure 8 the feature detector in 2D array.

As you can see in the previous Fig, each output value in the feature map does not have to connect to each pixel value in the input image. It only needs to connect to the receptive field, where the filter is being applied. Since the output array does not need to map directly to each input value, convolutional (and pooling) layers are commonly referred to as "partially connected" layers. However, this characteristic can also be described as local connectivity.

Note that the weights in the feature detector remain fixed as it moves across the image, which is also known as parameter sharing. Some parameters, like the weight values, adjust during training through the process of backpropagation and gradient descent. However, there are three hyperparameters which affect the volume size of the output that need to be set before the training of the neural network begins. These include:

- The **number of filters** affects the depth of the output. For example, three distinct filters would yield three different feature maps, creating a depth of three.
- **Stride** is the distance, or number of pixels, that the kernel moves over the input matrix. While stride values of two or greater is rare, a larger stride yields a smaller output.
- **Zero-padding** is usually used when the filters do not fit the input image. This sets all elements that fall outside of the input matrix to zero, producing a larger or equally sized output.

There are three types of padding:

- **Valid padding:** This is also known as no padding. In this case, the last convolution is dropped if dimensions do not align.
- **Same padding**: This padding ensures that the output layer has the same size as the input layer.
- **Full padding**: This type of padding increases the size of the output by adding zeros to the border of the input.

After each convolution operation, a CNN applies a Rectified Linear Unit (ReLU) transformation to the feature map, introducing nonlinearity to the model.

As we mentioned earlier, another convolution layer can follow the initial convolution layer. When this happens, the structure of the CNN can become hierarchical as the later layers can see the pixels within the receptive fields of prior layers. As an example, let's assume that we're trying to determine if an image contains a bicycle. You can think of the bicycle as a sum of parts. It is comprised of a frame, handlebars, wheels, pedals, et cetera. Each individual part of the bicycle makes up a lower-level pattern in the neural net, and the combination of its parts represents a higher-level pattern, creating a feature hierarchy within the CNN.[35]

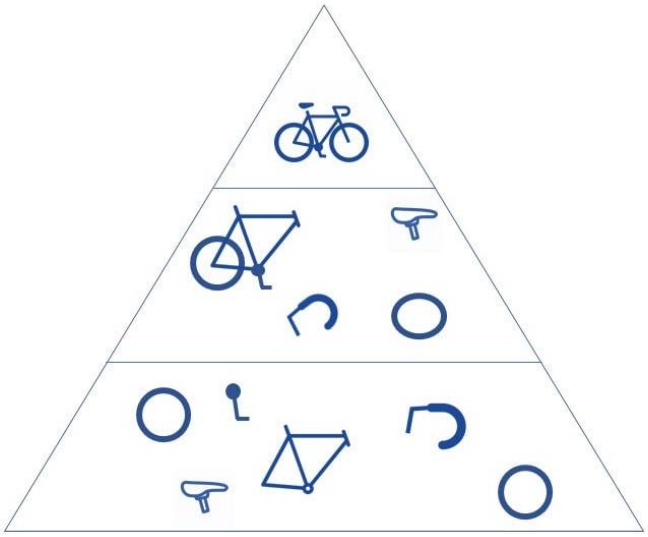

Figure 9 lower level and higher levels feature extractions.

Ultimately, the convolutional layer converts the image into numerical values, allowing the neural network to interpret and extract relevant patterns.

3.2. Pooling Layer

Pooling layers, also known as down sampling, conducts dimensionality reduction, reducing the number of parameters in the input. Like the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array. There are two main types of pooling:

- **Max pooling:** As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.
- **Average pooling:** As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.

While a lot of information is lost in the pooling layer, it also has a number of benefits to the CNN. They help to reduce complexity, improve efficiency, and limit risk of overfitting.

3.3. Fully-Connected Layer

The name of the full-connected layer aptly describes itself. As mentioned earlier, the pixel values of the input image are not directly connected to the output layer in partially connected layers. However, in the fully connected layer, each node in the output layer connects directly to a node in the previous layer.

This layer performs the task of classification based on the features extracted through the previous layers and their different filters. While convolutional and pooling layers tend to use ReLu functions, FC layers usually leverage a SoftMax activation function to classify inputs appropriately, producing a probability from 0 to 1.

3.4. Types of convolutional neural networks

Let's talk about different Types of CNN architectures:

- **LeNet**
- **AlexNet**
- **GoogLeNet**
- **VGGNet**
- **ResNet:** ResNet is the CNN architecture that was developed by Kaiming He et al. to win the ILSVRC 2015 classification task with a top-five error of only 15.43%. The network has 152 layers and over one million parameters, which is considered deep even for CNNs because it would have taken more than 40 days on 32 GPUs to train the network on the ILSVRC 2015 dataset. CNNs are mostly used for image classification tasks with 1000 classes, but ResNet proves that CNNs can also be used successfully to solve natural language processing problems like sentence completion or machine comprehension, where it was used by the Microsoft Research Asia team in 2016 and 2017 respectively. Real-life applications/examples of ResNet CNN architecture include Microsoft's machine comprehension system, which has used CNNs to generate the answers for more than 100k questions in over 20 categories. The CNN architecture ResNet is computationally efficient and can be scaled up or down to match the computational power of GPUs. [36]

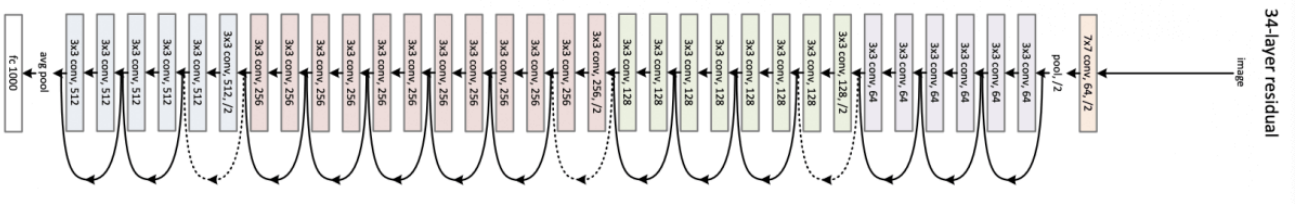

Figure 10 ResNet architecture.

- **MobileNets:** MobileNets are CNNs that can be fit on a mobile device to classify images or detect objects with low latency. MobileNets have been developed by Andrew G Trillion et al.. They are usually very small CNN architectures, which makes them easy to run in real-time using embedded devices like smartphones and drones. The architecture is also flexible so it has been tested on CNNs with 100-300 layers and it still works better than other architectures like VGGNet. Real-life examples of MobileNets CNN architecture include CNNs that is built into Android phones to run Google's Mobile Vision API, which can automatically identify labels of popular objects in images [36]

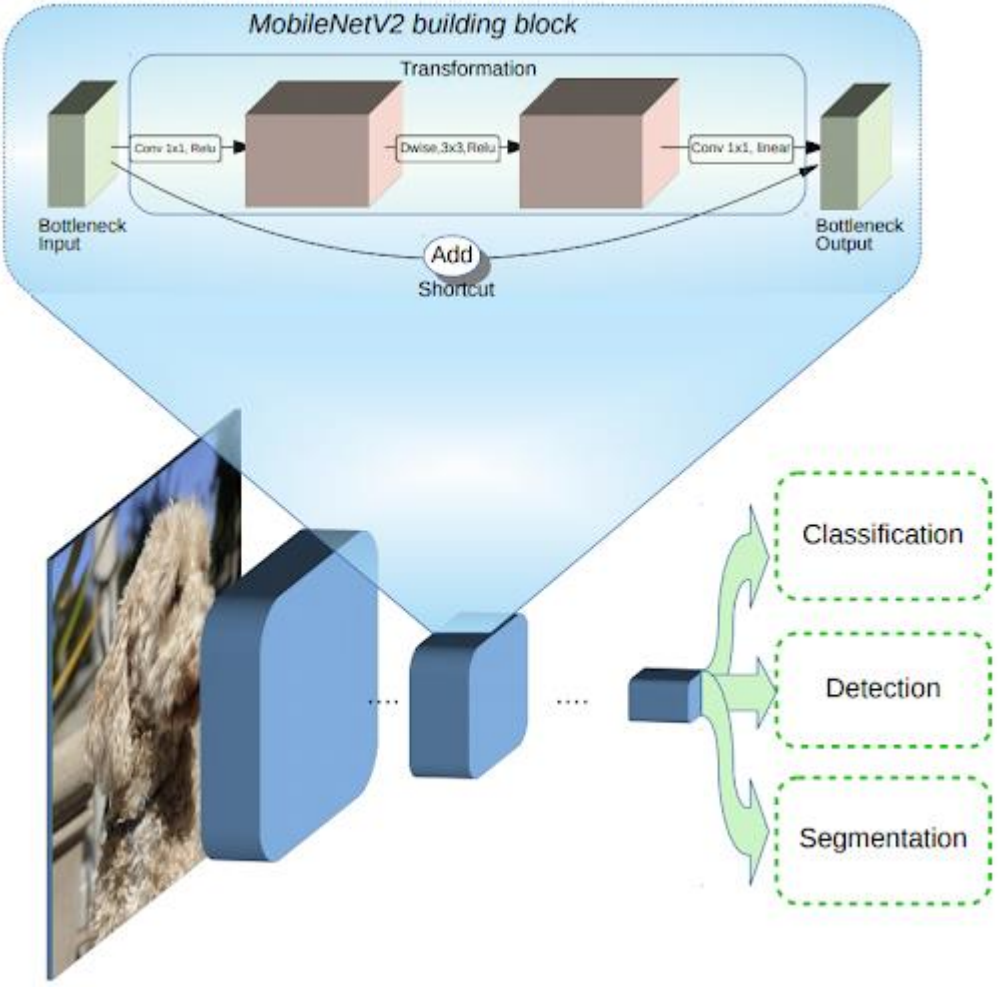

Figure 11 MobileNetV2 architecture.

3.5. Feature Extraction Using Light-Weight CNN

The core concept at backend of Mobile Net paradigms is to supplant huge and costly convolutional layers with depth-wise distinguishable convolutional blocks. These convolutional blocks consist of two key elements:

- a **depth-wise** convolutional layer that uses 3 × 3 filters for a given input
- **point-wise** convolutional layers that incorporate a 1 × 1 filter that functions to merge these filtered values and extract the learned features.

The MobileNet model is light-weight and considerably faster than a conventional convolution-based model and achieves approximately the same results. The MobileNetV1 has 3 × 3 convolution and 13 depth-wise distinguishable convolutional blocks [37].

MobileNetV2 contains one extra expansion layer in each block with a filter size of 1 × 1 for point-wise and depth-wise convolutional layers [38].

The core objective of this layer is to increase the number of channels in the data rather than moving to the depth-wise convolution. As a result, this layer generates more output channels than the given input channels. Unlike V1, the point-wise layers of MobileNetV2 are part of the projection layer: this layer is responsible for projecting data with huge number of channels into tensors, along with a small number of channels The main function of the bottleneck residual block is

that it provides the end result of these blocks; the residual block is modified, using convolutions to build a bottleneck. This block is valuable for reducing the number of parameters and matrix multiplications. As usual, every layer of MobileNetV2 contains ReLU6 and batch normalization, which is used as an activation function. The results of the projection layer are generated without using the activation function, but the overall composition of the MobileNet involves 17 bottleneck residual blocks and regular 1 × 1 convolution, followed by a global average pooling and classification layer. The top of MobileNetV2 is the global average pooling layer, which is helpful in reducing the problem of overfitting. [23]

The MobileNetV2 model is pretrained on the challenging ImageNet dataset, which consists of 1000 classes and approximately 1.4 M images, and we use these weights in our model.

We exclude the topmost layer of the MobileNetV2, which is ideal for feature extraction. We extracted features from a 30-frame sequence; these features are further passed through our proposed BD-LSTM to recognize anomalous activities.

Following Figure 2.Representation of typical, depth-wise, and pointwise convolution, where I, M, and N represent image, dimensions, and number of channels, respectively.

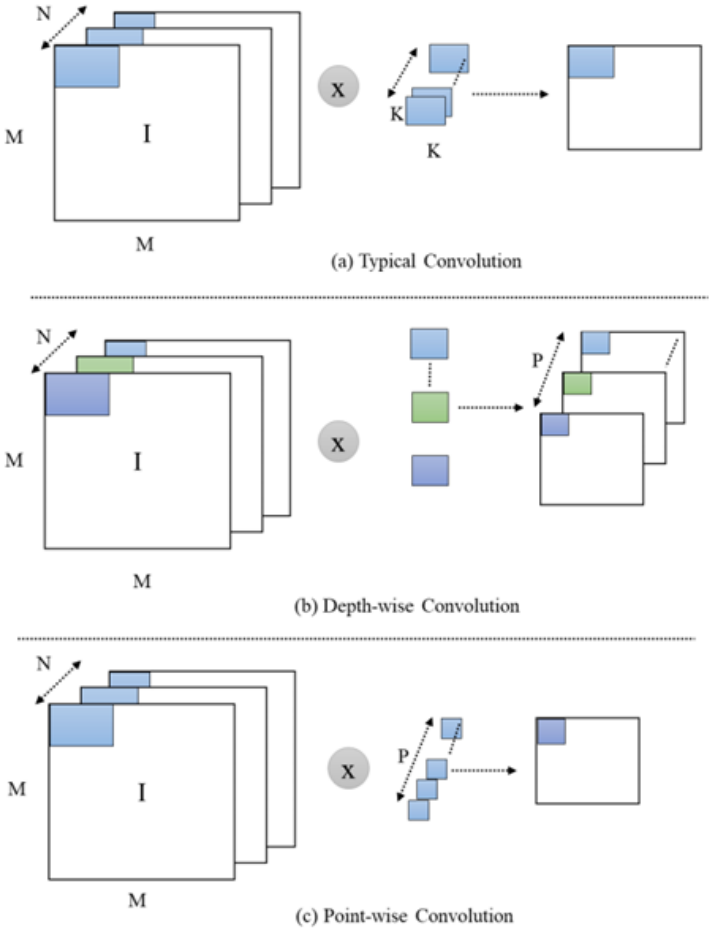

Figure 12 point-wise and Depth-wise convolution.

4. Module: Recurrent Neural Network RNN

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (NLP), speech recognition, image captioning and Video Classification.

they are incorporated into popular applications such as Siri, voice search, and Google Translate. Like feedforward and convolutional neural networks (CNNs), recurrent neural networks utilize training data to learn. They are distinguished by their "memory" as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. While future events would also be helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions.[41]

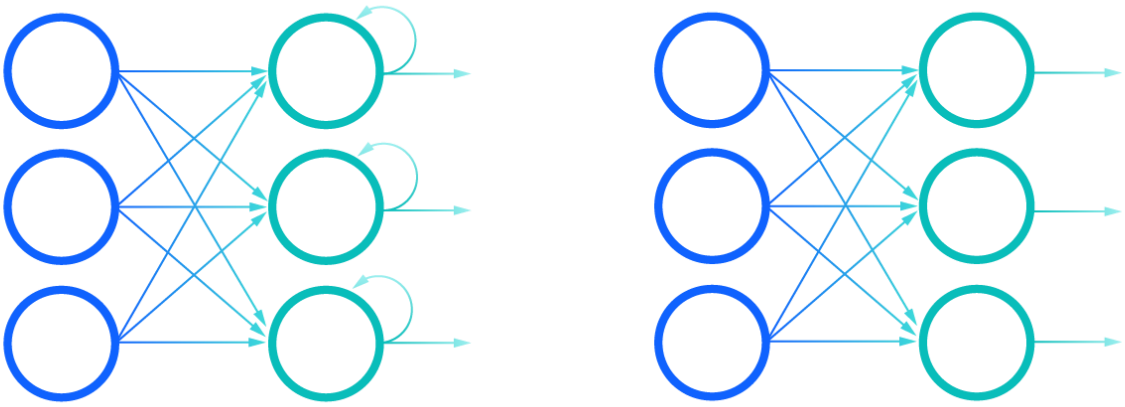

Figure 13 Recurrent neural network on the left versus feed forward neural network on the right.

Looking at the Fig. below, the "rolled" visual of the RNN represents the whole neural network, or rather the entire predicted Sequence.

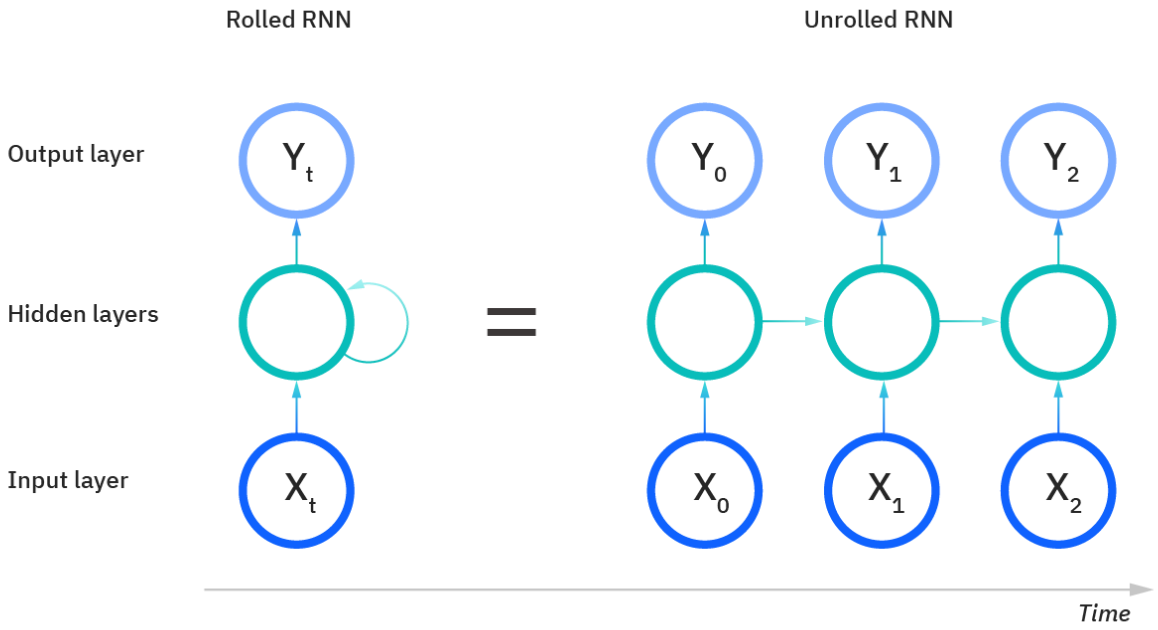

Figure 14 rolled RNN and unrolled RNN.

Another distinguishing characteristic of recurrent networks is that they share parameters across each layer of the network. While feedforward networks have different weights across each node, recurrent neural networks share the same weight parameter within each layer of the network. That said, these weights are still adjusted in the through the processes of backpropagation and gradient descent to facilitate reinforcement learning.

Also, one of the most important characteristics of the RNN is their inputs and outputs can vary in length as we will see in the coming section we will talk about that property and its effect on the types of RNN

Recurrent neural networks leverage backpropagation through time (BPTT) algorithm to determine the gradients, which is slightly different from traditional backpropagation as

it is specific to sequence data. The principles of BPTT are the same as traditional backpropagation, where the model trains itself by calculating errors from its output layer to its input layer. These calculations allow us to adjust and fit the parameters of the model appropriately. BPTT differs from the traditional approach in that BPTT sums errors at each time step whereas feedforward networks do not need to sum errors as they do not share parameters across each layer.

Through this process, RNNs tend to run into two problems known as

- **exploding gradients.**
- **vanishing gradients.**

These issues are defined by the size of the gradient, which is the slope of the loss function along the error curve. When the gradient is too small, it continues to become smaller, updating the weight parameters until they become insignificant—i.e. 0. When that occurs, the algorithm is no longer learning (Vanishing problem).

Exploding gradients occur when the gradient is too large, creating an unstable model. In this case, the model weights will grow too large, and they will eventually be represented as NaN. One solution to these issues is to reduce the number of hidden layers within the neural network, eliminating some of the complexity in the RNN model.

4.1. Types of recurrent neural networks

Feedforward networks map one input to one output, and while we've visualized recurrent neural networks in this way in the above diagrams, they do not actually have this constraint. Instead, their inputs and outputs can vary in length, and different types of RNNs are used for different use cases, such as music generation, sentiment classification, and machine translation. Different types of RNNs are usually expressed using the following diagram:

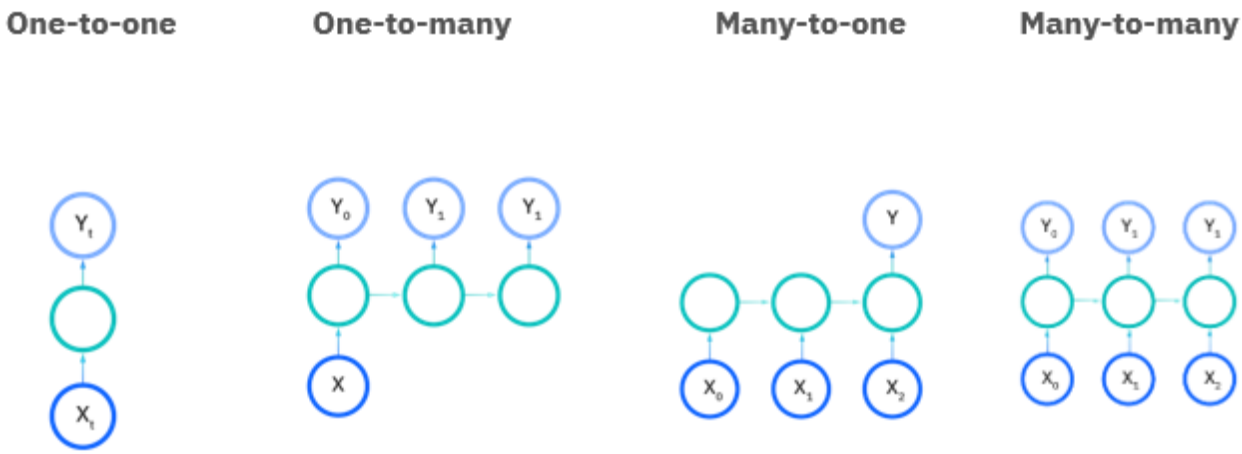

Figure 15 RNN different Types.

4.2. Variant RNN architectures

4.2.1. Long short-term memory (LSTM):

This is a popular RNN architecture, which was introduced by Sepp Hochreiter and Juergen Schmidhuber as a solution to vanishing gradient problem. That is, if the previous state that is influencing the current prediction is not in the recent past, the RNN model may not be able to accurately predict the current state. As an example, let's say we wanted to predict the italicized words in following, "Alice is allergic to nuts. She can't eat *peanut butter*." The context of a nut allergy can help us anticipate that the food that cannot be eaten contains nuts. However, if that context was a few sentences prior, then it would make it difficult, or even impossible, for the RNN to connect the information. To remedy this, LSTMs have "cells" in the hidden layers of the neural network, which have three gates–an input gate, an output gate, and a forget gate. These gates control the flow of information which is needed to predict the output in the network. For example, if gender pronouns, such as "she", was repeated multiple times in prior sentences, you may exclude that from the cell state.

But How to Solve the Vanishing and exploding problem This is the main reason of why there are anther versions from simple RNN like LSTM and GRUs which solve that vanishing problem and let's talk about LSTM in more details

Sequential Learning Techniques LSTM was introduced to resolve the vanishing or exploding gradients issue in recurrent neural networks, and involves internal memory cells that are controlled by an input and forget gate network. The cell state is altered by the forget gate ranked under the cell state and also modified by the input gate. Additionally, the main purpose of the forget gate in the LSTM is to decide how much information from the previous memory should be passed into the next time step. Similarly, the input gate first regulates how much new information should be entered into the memory cell and then a vector is formed applying the tan h function, which provides output. Depending on these gates, LSTM can handle the short- and long-term dependency of the sequential information [40].

Its chain like building block with forget, input, and output gates, can regulate long-term sequence pattern recognition. The sigmoid function units are the main component of these gates, which acquires over the duration of training as it is open and closed. The dataflow and operations are processed by the LSTM unit from the input to the output gates. In the Equations, " t" is the input over time "t", the sigmoid function is represented by "σ", "ω"and"ß" are the weights and bias terms of the training stage, respectively [39]. "it "" t" and "Ot " represent the three main gates of the LSTM at time "t" (input, forget and output respectively). The input gate "it" monitors when to record the current input " t", and the forget gate " t" determines when to release the preceding memory cell "Ct−1 ". The output gate "Ot verifies the data shifted from current memory "Ct " to the hidden state. In this state, the step is analyzed using "tanh" activation and a memory cell "Ct ". As anomaly detection does not require the transitional output of the LSTM, the final determnation is made through the SoftMax classifier in the prediction state of the RNN.[21]

The LSTM formulation is expressed as follows:

$$i_t = \sigma(\omega_i[X^t + Ş_{t-1}] + ß_i)$$

$$F_t = \sigma(\omega_F[X^t + Ş_{t-1}] + ß_F)$$

$$\bar{O}_t = \sigma(\omega_{\bar{O}}[X^t + Ş_{t-1}] + ß_{\bar{O}})$$

$$R = tanh(\omega_R[X^t + Ş_{t-1}] + ß_R)$$

$$\acute{C}_t = \acute{C}_{t-1} \cdot F_t + R \cdot i_t$$

$$Ş_t = tanh(\acute{C}_t) \cdot \bar{O}_t$$

$$Prediction_{state} = SoftMax(V_{Ş_t})$$

Figure 16 LSTM basic equations.

- The **single LSTM** cell cannot classify large amounts of training data such as the dynamic sequence patterns in videos. We therefore created a multi-layer-LSTM by assembling several LSTM cells to efficiently understand long-term sequence dependencies. The internal structure of an LSTM is presented in Fig. 3.

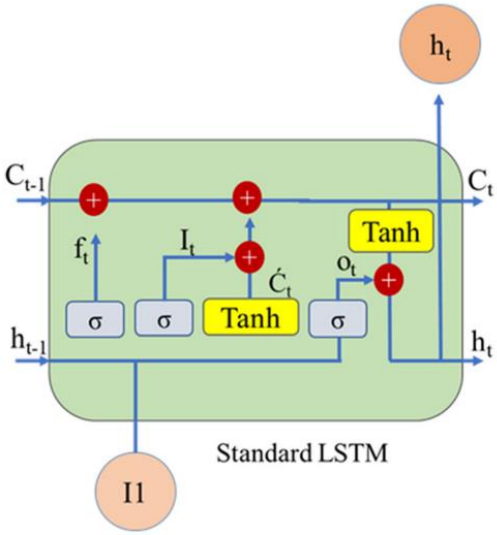

Figure 17 standard single LSTM.

- **Multi-layer LSTM**

The efficiency of a deep neural network depends upon the number of layers. A similar approach is pursued here for an RNN by stacking our network with two LSTMs, resulting in the learning of high-level sequence information [39].

In a typical LSTM for activation and processing, the data is transferred to a single layer prior to output however, in time sequence problems, we require to analyze information across multiple layers. Each layer in the LSTM is often a hierarchy that accepts the hidden state of the preceding layer as input via multiple LSTM layers, as shown in Fig. 4. The first layer of the LSTM, receives input sequential information as t, whereas the given input to the second layer is received from the preceding time step and the output from the first layer is . The LSTM cell calculation is similar to Eqs. 1 to 7 except that the information from each layer, it, t, Ot ,Ct and is applied to every other layer. The process used to calculate the layer state is shown in Eq. 8.[21]

$$Ş_t^l = tanh(\acute{C}_t{}^l) \cdot \bar{O}_t{}^l$$

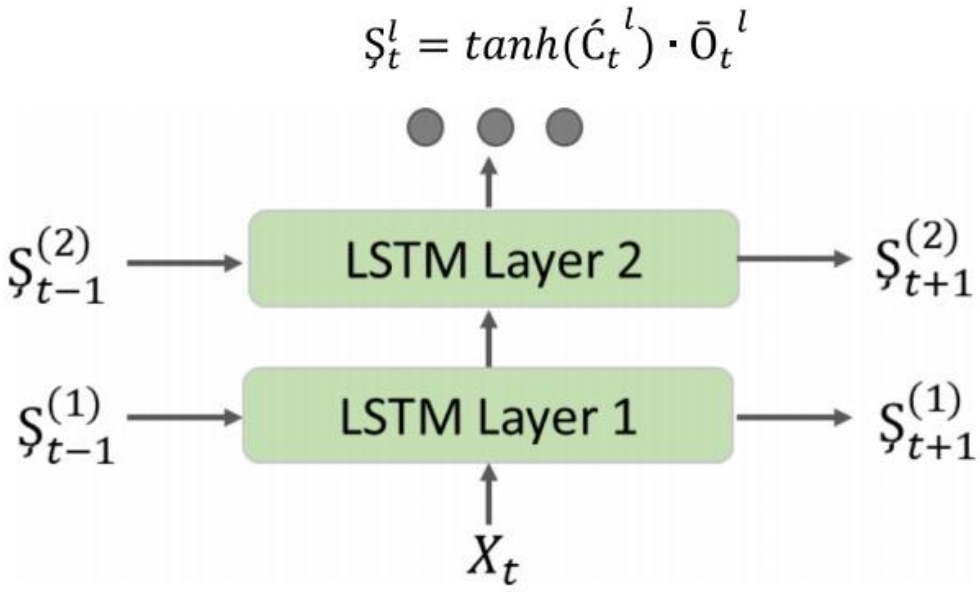

Figure 18 Multi-layer LSTM architecture.

- **Bi-directional LSTM**

Unlike a simple LSTM, the output of the BD-LSTM depends not only on the previous frames but also on the upcoming frames in the sequence. The structure of the BD-RNN is relatively simple with two stacked RNNs, one in backwards direction and the other in a forward direction, while the hidden states of both RNNs are combined in the output. In this work, we used a multi-layer BD-LSTM in which each layer has two cells, one for the backwards pass and the other for the forward pass. [21]

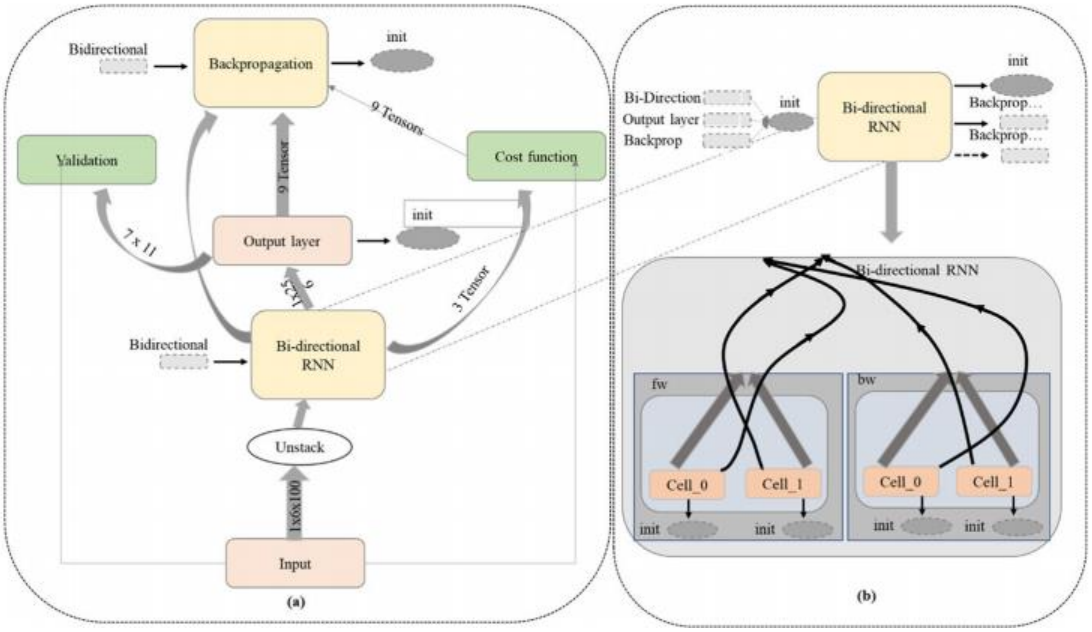

Figure 19 BD-LSTM architecture.

1.1.1. Gated recurrent units (GRUs):

This RNN variant is like the LSTMs as it also works to address the short-term memory problem of RNN models. Instead of using a "cell state" regulate information, it uses hidden states, and instead of three gates, it has two—a reset gate and an update gate. Like the gates within LSTMs, the reset and update gates control how much and which information to retain.

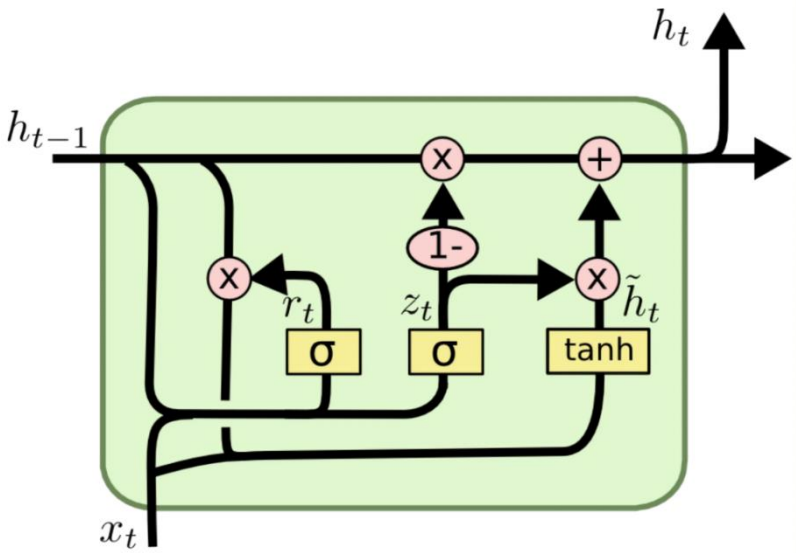

Figure 20 Gated recurrent units.

5. Module: Dataset

5.1. Dataset Description

The dataset used in our S3CB Project is UCF-Crime dataset which cover 13 real world anomalies, including Abuse, Arrest, Arson, Assault, Road Accident, Burglary, Explosion, Fighting, Robbery, Shooting, Stealing, Shoplifting, and Vandalism.

There are Short Descriptions about each type of anomaly in the General dataset according to the reference [1] as follows:

- **Abuse**: This event contains videos which show bad, cruel, or violent behavior against children, old people, animals, and women.

- **Burglary**: This event contains videos that show people (thieves) entering into a building or house with the intention to commit theft. It does not include use of force against people.

- **Robbery**: This event contains videos showing thieves taking money unlawfully by force or threat of force. These videos do not include shootings.

- **Stealing**: This event contains videos showing people taking property or money without permission. They do not include shoplifting.

- **Shooting**: This event contains videos showing act of shooting someone with a gun.

- **Shoplifting**: This event contains videos showing people stealing goods from a shop while posing as a shopper.

- **Assault**: This event contains videos showing a sudden or violent physical attack on someone. Note that in these videos the person who is assaulted does not fight back.

- **Fighting**: This event contains videos displaying two are more people attacking one another.
- **Arson**: This event contains videos showing people deliberately setting fire to property.
- **Explosion**: This event contains videos showing destructive events of something blowing apart. This event does not include videos where a person intentionally sets a fire or sets off an explosion.
- **Arrest**: This event contains videos showing police arresting individuals.
- **Road Accident:** This event contains videos showing traffic accidents involving vehicles, pedestrians, or cyclists.
- **Vandalism**: This event contains videos showing action involving deliberate destruction of or damage to public or private property. The term includes property damage, such as graffiti and defacement directed towards any property without permission of the owner.
- **Normal Event:** This event contains videos where no crime occurred. These videos include both indoor (such as a shopping mall) and outdoor scenes as well as day and night-time scenes.

We focused on the **Indoor Anomalies** and Focus on the most common anomalies used such as Fighting, Explosions, Robbery, Shooting and Shoplifting And we Found a lot of draw backs while working with the dataset we will discuss it in details.

The drawbacks of the dataset are:

- Missing videos.
- Unclear videos.
- Repeated videos.
- Videos with wrong Label.

5.2. Dataset review and Classifications

In this approach dataset was reviewed video by video for each anomaly to analyze what the nature of its content and classify similar videos based on its description, and by the outcome information decisions were made what type of videos that would be useful and what is the outliers.

Assault:

#Case	Description	Number of Videos	
		Out	In
1	Group of people assault with their hands & legs.	15	5
2	Group of people assault with sticks.	6	1
3	One person assaults another person.	9	10
4	One person threatens another one with gun.	1	-
5	push someone on the train.	1	-
6	Assault with chair & stick.	-	0
7	Verbal assault.	-	0

Table 3 Assault Cases Description

Burglary:

#Case	Description	Number of Videos	
		Out	In
1	break into a house/store through the door.	13	39
2	break into a house through the Window.	4	8
3	Burgling an ATM machine using Car.	3	8
4	Burgle something and put it in the car.	7	2
5	break into a house by jumping over the fence.	1	-
6	Stealing a chicken skewer	1	-
7	Broken glass door.	1	-
8	Try to burgle a cashier	-	2
9	break a safe	-	2

Table 4 Burglary Case Description

Shoplifting:

#Case	Description	Number of Videos	
		Out	In
1	Someone hides a product from a store in his clothes / bag.	-	50

Table 5 Shoplifting Case Description

Explosion:

#Case	Description	Number of Videos	
		Out	In
1	Explosion at end of the road.	5	-
2	Explosion at gas station.	7	-
3	Car exploded.	13	-
4	House door exploded while fire men try to open it to enter.	1	-
5	Bomb explosion.	11	4
6	Home Gas leak explosion.	-	1
7	Electric device Explosion.	2	5
8	Warehouse explosion.	-	2
9	Large Factory Explosion.	1	-

Table 6 Explosion Case Description

Stealing:

#Case	Description	Number of Videos	
		Out	In
1	Stealing a car.	8	4
2	Stealing a Motor Cycle.	16	6
3	Failed attempts of stealing a car, or something from a car.	34	12
4	Steal something from a motor Cycle. (helmet/spare tier)	3	1
5	Steal something Infront of a shop and run away.	1	-
6	Stealing some one's Bag.	-	3
7	Stealing mobile phone from the money drawer at the shop.	-	1
8	Stealing something form a garden.	1	3
9	Un related Videos.	7	-

Table 7 Stealing Case Description

Shooting:

#Case	Description	Number of Videos	
		Out	In
1	One person shooting.	8	2
2	Intense Argument leads to gang shooting.	5	-
3	Multiple people shooting each other.	6	-
4	Police man shoot some criminal.	6	2
5	Someone breaks into a store and shoot someone.	1	7
6	Fire exchange between officers and criminals.	2	3
7	Shoot for self-defense against robbery.	3	3
8	Un related	1	1

Table 8 Shooting Case Description

Fighting:

#Case	Description	Number of Videos	
		Out	In
1	Group of people Fighting using their hands and legs.	12	15
2	Group of people fight with sticks.	1	-
3	One person Fights with another person.	9	11
4	Fight using chair.	-	1

Table 9 Fighting Case Description

Robbery:

#Case	Description	Number of Videos	
		Out	In
1	Armed people threaten people and robbed their property.	13	126
2	A person riding Motorcycle snatch someone's property.	5	-
3	Shooting.	-	1
4	Fighting.	-	2
5	Assault.	-	1

Table 10 Robbery Case Description

We filtered our dataset by removing Missing videos and also unclear videos and also repeated videos and put videos that placed in the Wrong label to Videos with Right Places.

Category	#videos	#used Videos	#Excluded Videos
Explosion	52	44	8
Fighting	50	41	9
Shooting	54	46	8
Shoplifting	50	45	5
Robbery	150	114	36
Vandalism	50	28	22
Burglary	100	20	80

Table 11 Filtered Dataset

5.3. Dataset Annotation

5.3.1. Explosion Annotation Samples:

Index	Video Name	Total Frames	Start Frame	End Frame	Used Frames	Saved Frames
0	Explosion001	693	200	635	435	258
1	Explosion002	4013	1492	1831	339	3674
2	Explosion003	576	118	170	52	524
3	Explosion004	1902	93	200	107	1795
4	Explosion005	693	244	683	439	254
	Total	**102160**	**0**	**0**	**18477**	**83683**

Table 12 Explosion Annotation

5.3.2. Shooting Annotation Samples:

Index	Video Name	Total Frames	Start Frame	End Frame	Used Frames	Saved Frames
0	Shooting001	253	157	195	38	215
1	Shooting003	2277	611	930	319	1958
2	Shooting004	1793	486	555	69	1724
3	Shooting005	5330	810	936	126	5204
4	Shooting006	13073	8990	9030	40	13033
	Total	**135384**	**0**	**0**	**10349**	**125035**

Table 13 Shooting Annotation

5.3.3. Shoplifting Annotation Samples:

	Video Name	Total Frames	Start Frame	End Frame	Used Frames	Saved Frames
0	Shoplifting003_x264.mp4	10817	7250	7420	170	10647
1	Shoplifting005_x264.mp4	1967	760	845	85	1882
2	Shoplifting006_x264.mp4	3156	1600	1740	140	3016
3	Shoplifting008_x264.mp4	8927	4375	5065	690	8237
4	Shoplifting009_x264.mp4	5201	4375	5050	675	4526
Σ	**Total**	**221300**	**0**	**0**	**25988**	**195312**

Table 14 Shoplifting Annotation

5.3.4. Fighting Annotation Samples:

	Video Name	Total Frames	Start Frame	End Frame	Used Frames	Saved Frames
0	Fighting003_x264.mp4	3102	1830	3101	1271	1831
1	Fighting004_x264.mp4	16777	45	15920	15875	902
2	Fighting005_x264.mp4	1784	1140	1385	245	1539
3	Fighting006_x264.mp4	944	310	725	415	529
4	Fighting007_x264.mp4	3794	1300	2720	1420	2374
Σ	**Total**	**202119**	**0**	**0**	**45703**	**156416**

Table 15 Fighting Annotation

5.3.5. Robbery Annotation Samples:

	Video Name	Total Frames	Start Frame	End Frame	Used Frames	Saved Frames
0	Robbery001_x264.mp4	987	535	755	220	767
1	Robbery002_x264.mp4	3203	245	441	196	3007
2	Robbery003_x264.mp4	984	259	594	335	649
3	Robbery004_x264.mp4	1529	223	1143	920	609
4	Robbery005_x264.mp4	2759	276	855	579	2180
Σ	**Total**	**336963**	**0**	**0**	**146638**	**190325**

Table 16 Robbery Annotation

5.3.6. Vandalism Annotation Samples:

	Video name	Total frames	Start frame	End frame	Used frames	Saved frames
1	Vandalism008_x264	12589	3104	3386	282	12307
2	Vandalism010_x264	3945	2178	2276	98	3847
3	Vandalism011_x264	12449	1059	1199	140	12309
Σ	**Total**	**97431**	**0**	**0**	**4628**	**92749**

Table 17 Vandalism Annotation

5.3.7. Burglary Annotation Samples:

	Video name	Total frames	Start frame	End frame	Used frames	Saved frames
1	Burglary001_x264	3969	1059	1132	73	3896
2	Burglary005_x264	7729	4733	4808	75	7654
3	Burglary007_x264	1085	833	946	113	972
	Total	**138973**	**0**	**0**	**3388**	**135585**

Table 18 Burglary Annotation

5.3.8. Annotation Summary

Then we go to Annotate our dataset to focus on the actions in an accurate way

So, we focused on how to make efficient use of our dataset with our Algorithm So the First Method we follow is Data Annotation to the classes we are used

Category	Total Frames	Total Used Frames	Total Saved Frames	Avg. Used Frames	Avg. Saved Frames	%Of Used Frames	%Of Saved Frames
Explosion	102160	18477	83683	18	81	18	82
Fighting	202119	44508	155282	440	2548	23.2	76.8
Shooting	135384	10349	125035	8	92	7.65	92.35
shoplifting	221300	25988	195312	288	2354	11.75	88.25
Robbery	336963	146638	190325	740	985	43.52	56.48
Vandalism	97431	4628	92749	167.21	3312.46	4.75	95.25
Burglary	138973	3388	135585	169.4	6779.25	2.48	97.52

Table 19 Annotation Summary

We have started our Model on the Five Cases after Annotation but We Found There are A great Conflict between Robbery and Fighting and that is really difficult to distinguish between these two classes as there are great similarity between these both classes Except in some videos in Robbery Class There is no certain pattern in that class as many thefts Carry gun with a quite manner and take the money or the goods in a very calm way as the owner of the Market is afraid because of that gun so no Reaction occur so that pattern to detect it is so similar to the normal case and difficult to recognize it and that's the most of the videos of Robbery case (Robbery Total videos = 150 after filtering found 114 videos we found 89 out of 114 is very quiet and similar to Normal class and the remaining 25 videos there is a resist from the supermarket owners and very similar to the Fighting Case) so we Gather the Both classes together Fighting and 25 videos of the Robbery To form a new class under the Umbrella of Fighting class.

The second issue that we found that the shoplifting class is so difficult as we see each video from 50 videos and 44 videos annotated and 6 Excluded But all the 44 video is so difficult to recognize as they are so like the Normal Case.

Shooting class contains 50 videos but after factorization we collect 46 videos and annotated them, But Still the Model has not the ability to recognize the pattern of shooting as it is so general and there are another object detection models can be more efficient for detecting weapons.

Explosion class is the one of the most distinctive classes we have used 43 videos out of 50 total videos and, we have annotated it.

So, we have used it and after conclusion we use in our dataset the cases that is general for the most cases occur which are:

Normal case we have used 150 videos

Explosion case we have used 43 videos

Fighting and Robbery case we have used 69 videos

The number of videos is not enough to train the deep learning model so data augmentation is the best solution to increase the number of videos.

5.4. Data Augmentation

Above of that as we have filtered and classify our dataset to indoor and outdoor as our system will focus on the indoor videos as S^{3}CB(Smart Surveillance System for Criminal Behavior)[our proposed system]so a lot of videos have been ignored so number of videos is not enough to train the Deep Learning Model so data augmentation is the best solution to increase number of videos using library imgaug[42]

Then we go to Increase our Dataset Using Data Augmentation Technique and we have tested many methods in data Augmentation like [42]:

5.4.1. Multiply with Brightness

Figure 21 Multiplied Brightness Augmentation

5.4.2. Flipping Videos 180 degree in a horizontal way

Figure 22 Flipped Horizontally Augmentation

5.4.3. Multiply with Brightness and flipping 180 degrees horizontally

Figure 23 Combined Multiplied Brightness and Flipped Horizontally Augmentation

5.4.4. Darkness and flipping Videos 180 degree in a horizontal way

Figure 24 Combined Darkening and Flipped Horizontally Augmentation

5.4.5. Adding Poisson Noise and flipping 180 in a horizontal way

Figure 25 Combined Added Poisson Noise and Flipped horizontally Augmentation

5.4.6. Using Dropout Technique and Multiply with Brightness

Figure 26 Combined Dropout and Multiplied Brightness Augmentation

5.4.7. Adding Gaussian noise

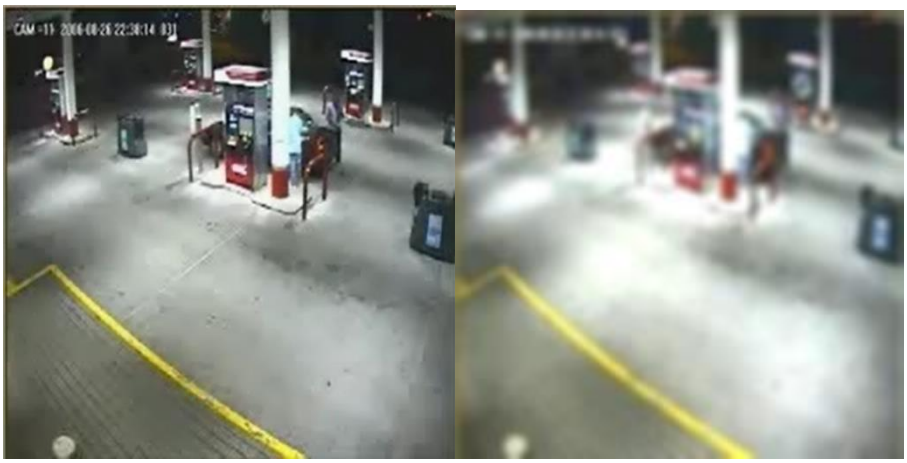

Figure 27 Adding Gaussian Noise Augmentation

6. Proposed Solution (Conclusion)

6.1. Hybrid model between CNN[MobileNetV2] + RNN [Multilayer BD-LSTM]

The features extracted by the MobileNetV2 are used to decide which anomalous events took place that are fed to the multi-layer BD-LSTM in the form of chunks for an anomaly recognition decision. The first chunk of the 1000 features from the initial frame of the video forms the input to the multi-layer BD-LSTM at time "t", while the next feature's chunk at t + 1 forms the second input to the multi-layer BD-LSTM, and so on. The overall structural design of the proposed multi-layer BD-LSTM is illustrated

shows the training stage, in which the training data are passed to the model.

The hidden state combines the forward and backward passes in the output layer, while backpropagation is used to adjust the bias and weights.

A sample of 20% of the total data is used for validation purposes, while cross-entropy is employed for error rate assessment along with a learning rate as default of 0.001 using cost minimization

Adam optimization. The interior structure of the BD-RNN where the forward and backward passes are represented as "fw" and "bw", respectively. The method of computing the output allows the proposed model to accomplish a high accuracy. The output of the frame is calculated based on the previous and upcoming frames, since each layer performs processing in both directions.

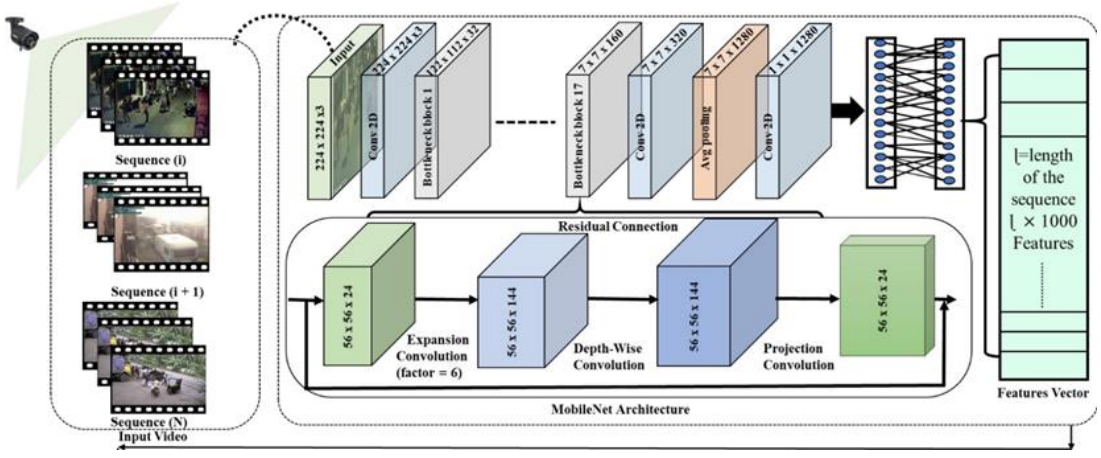

Figure 28 MobileNetV2 for extracting features of video images to be represented in a feature vector.

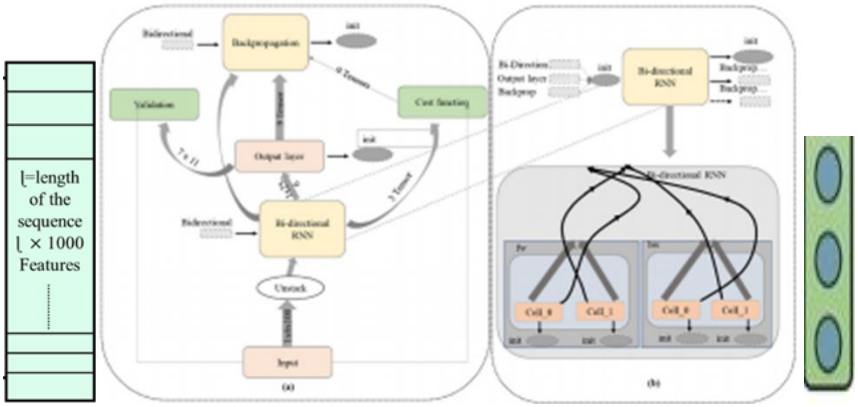

Figure 29 feature vector is fed to multi-layer BD-LSTM for temporal feature extraction and for prediction.

6.2. Proposed Model Structure:

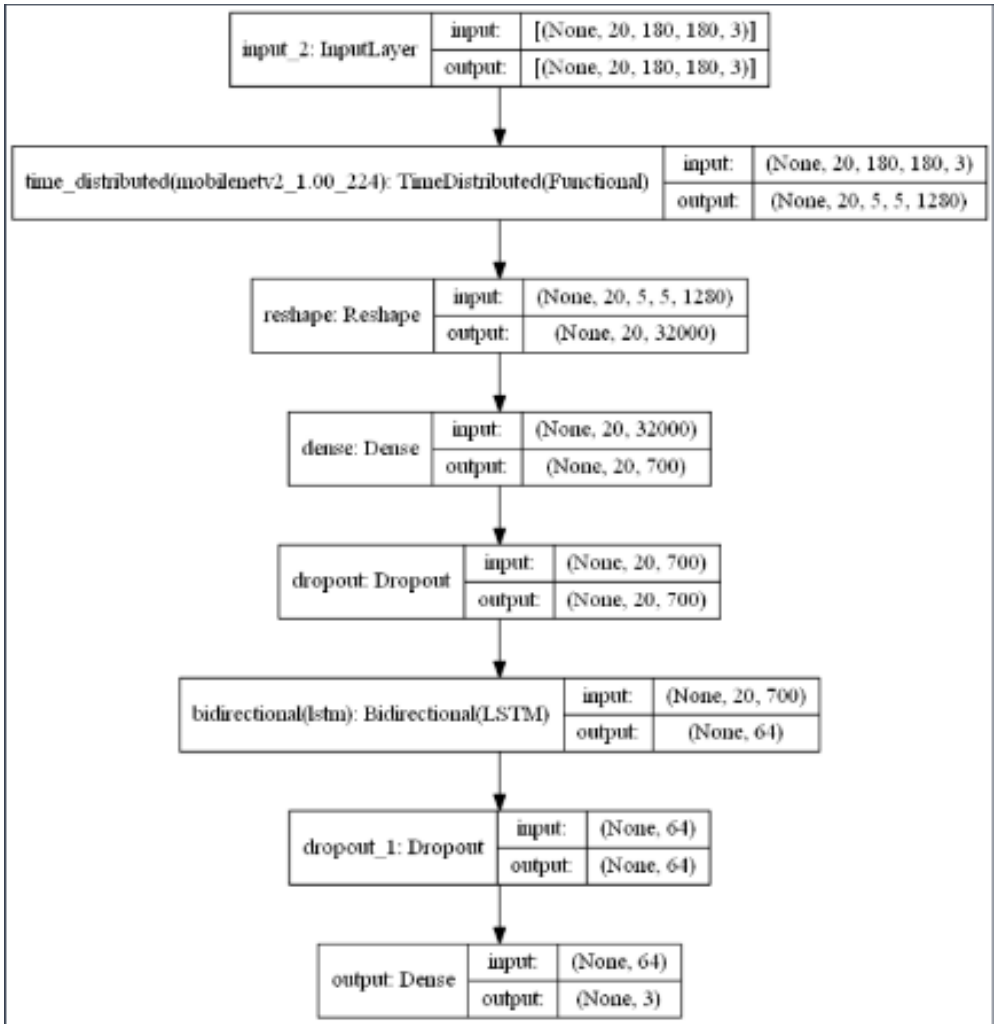

Figure 30 proposed model structure.

Conclusion after training our proposed model with the different types of data augmentation we reached the best type can be used to in data augmentation avoiding overfitting is flipping 180 degree and the last step we have taken to increase number of videos we used our annotation but for dividing each video to number of short videos that increase the accuracy of our model to reach to the final results as follows:

Metric	Measured Value
Recall	93.93%
Accuracy	93.03%
Precision	94.34%
AUC	99.15%

Table 20 Model Evaluation Metrics

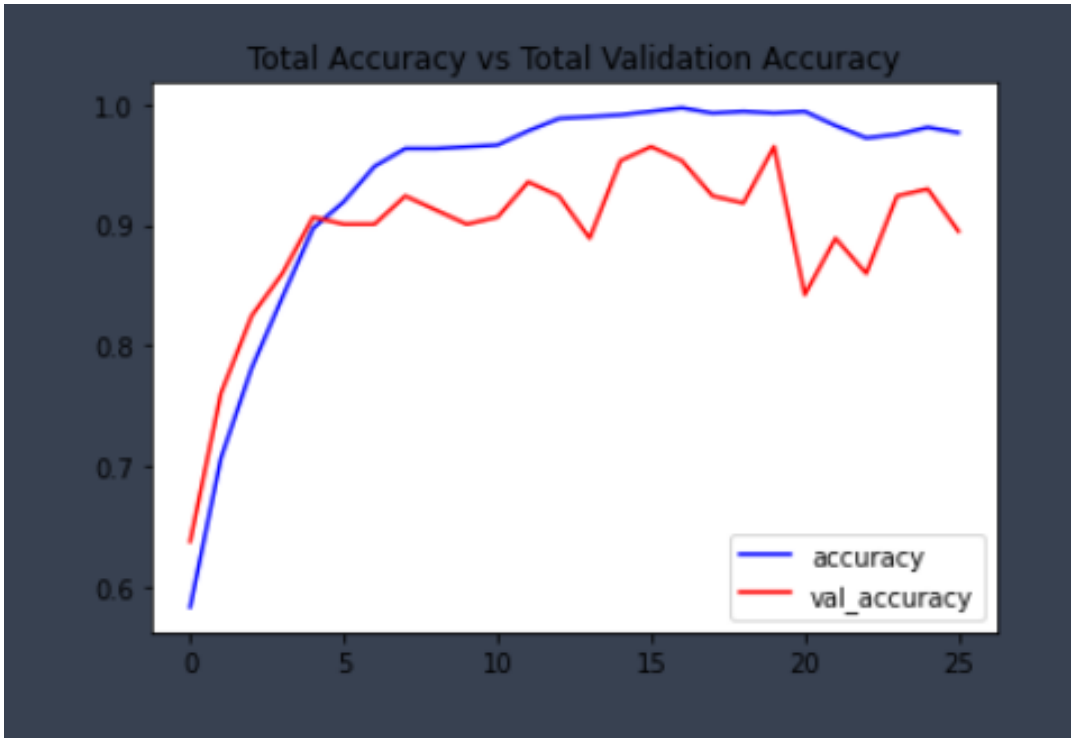

Figure 31 accuracy and validation accuracy of the proposed model versus number of epochs.

6.3. System block diagram

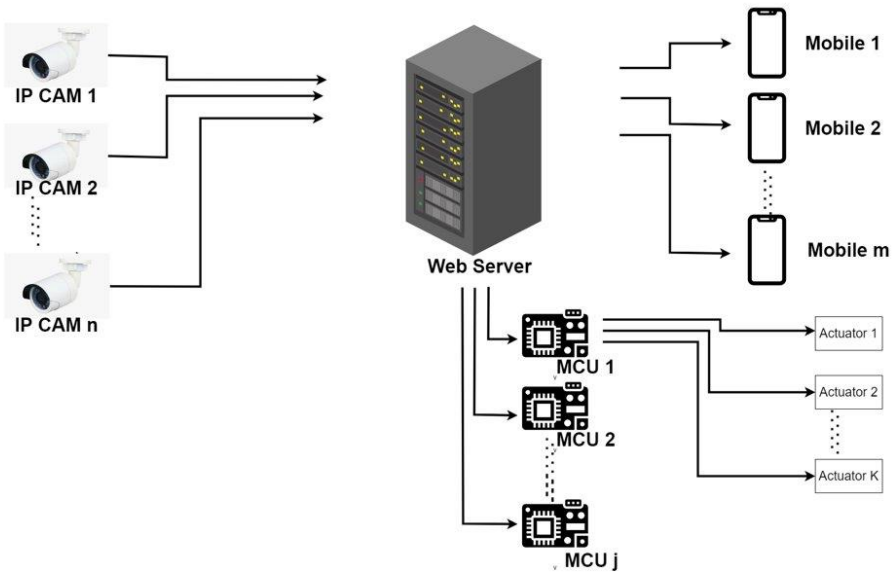

Figure 32 System Block Diagram

6.4. Proposed System Structure

The control room is the brain of the system; in which all operations are managed by it and all the communication operations between the various parts of the systems.

The software part of the system (control room) is consisting of:

1- Application programming interface (API).

2- SQL Server Database.

3- Mobile Application.

4- Admin website.

5- Stream of the cameras.

6- Local hosting on IIS Express.

7- Firebase DB for notifications and actions.

Each one of them will be discussed in details in this chapter.

Following Diagram will explain the communications between the different parts of the system and the API:

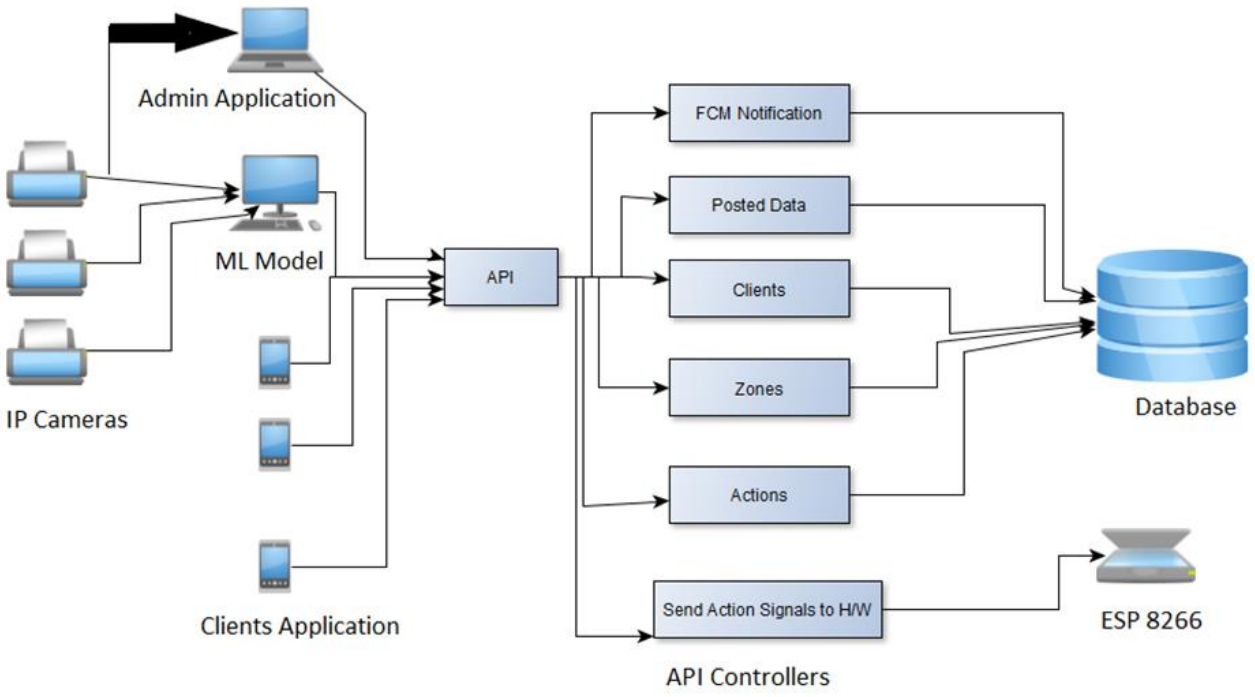

Figure 33 System API Diagram

7. Module: Application programming interface (API)

7.1. What is an API?

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. For example, the weather bureau's software system contains daily weather data. The weather app on your phone "talks" to this system via APIs and shows you daily weather updates on your phone.

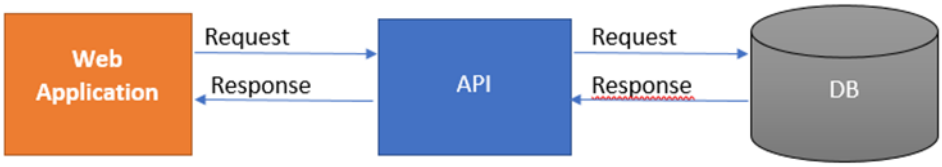

Figure 34 API Block Diagram

7.2. What does API stand for?

API stands for Application Programming Interface. In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses. Their API documentation contains information on how developers are to structure those requests and responses.

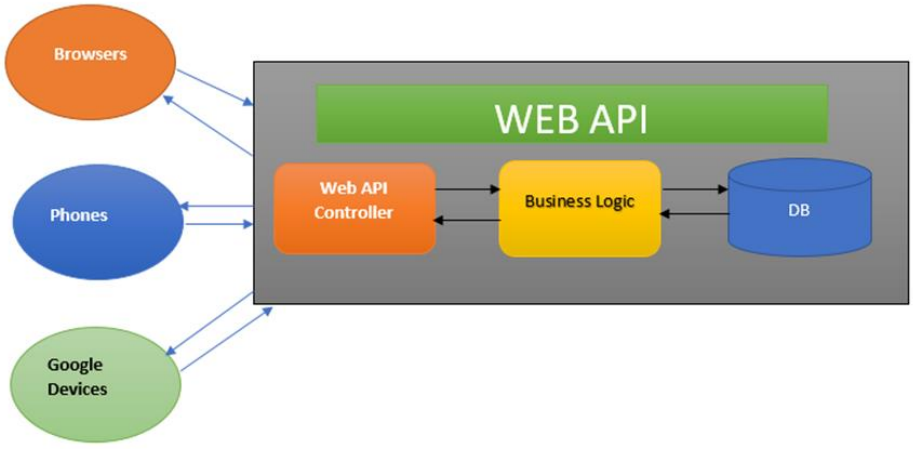

Figure 35 Web API Architecture diagram

This diagram explains the architecture of Web API.

- A client called API/controller – In the above diagram Browers, Phones, and Google Devices are called Web API Controllers.
- API/Controller interact with business layer and get Data from DB.
- The output will be returned in JSON format.

7.3. REST APIs

These are the most popular and flexible APIs found on the web today. The client sends requests to the server as data. The server uses this client input to start internal functions and returns output data back to the client. Let's look at REST APIs in more detail below.

7.4. What are REST APIs?

REST stands for Representational State Transfer. REST defines a set of functions like GET, PUT, DELETE, etc. that clients can use to access server data. Clients and servers exchange data using HTTP.

The main feature of REST API is statelessness. Statelessness means that servers do not save client data between requests. Client requests to the server are similar to URLs you type in your browser to visit a website. The response from the server is plain data, without the typical graphical rendering of a web page.

1.4. benefits REST APIs benefits:

REST APIs offer four main benefits:

1.4.1. Integration

APIs are used to integrate new applications with existing software systems. This increases development speed because each functionality doesn't have to be written from scratch.

You can use APIs to leverage existing code.

1.4.2. Innovation

Entire industries can change with the arrival of a new app. Businesses need to respond quickly and support the rapid deployment of innovative services. They can do this by making changes at the API level without having to re-write the whole code.

1.4.3. Expansion

APIs present a unique opportunity for businesses to meet their clients' needs across different platforms. For example, maps API allows map information integration via websites, Android, IOS, etc. Any business can give similar access to their internal databases by using free or paid APIs.

1.4.4. Ease of maintenance

The API acts as a gateway between two systems. Each system is obliged to make internal changes so that the API is not impacted. This way, any future code changes by one party do not impact the other party.

1.5. REST API Security

All APIs must be secured through proper authentication and monitoring. The two main ways to secure REST APIs include:

1.5.1. Authentication tokens

These are used to authorize users to make the API call. Authentication tokens check that the users are who they claim to be and that they have access rights for that particular API call. For example, when you log in to your email server, your email client uses authentication tokens for secure access.

1.5.2. API keys

API keys verify the program or application making the API call. They identify the application and ensure it has the access rights required to make the particular API call. API keys are not as secure as tokens but they allow API monitoring in order to gather data on usage. You may have noticed a long string of characters and numbers in your browser URL when you visit different websites. This string is an API key the website uses to make internal API calls.

7.5. API Operations

Web API is mostly used for CRED (Create, Read, EDIT, DELETE) operations. It follows HTTP verbs for these operations.

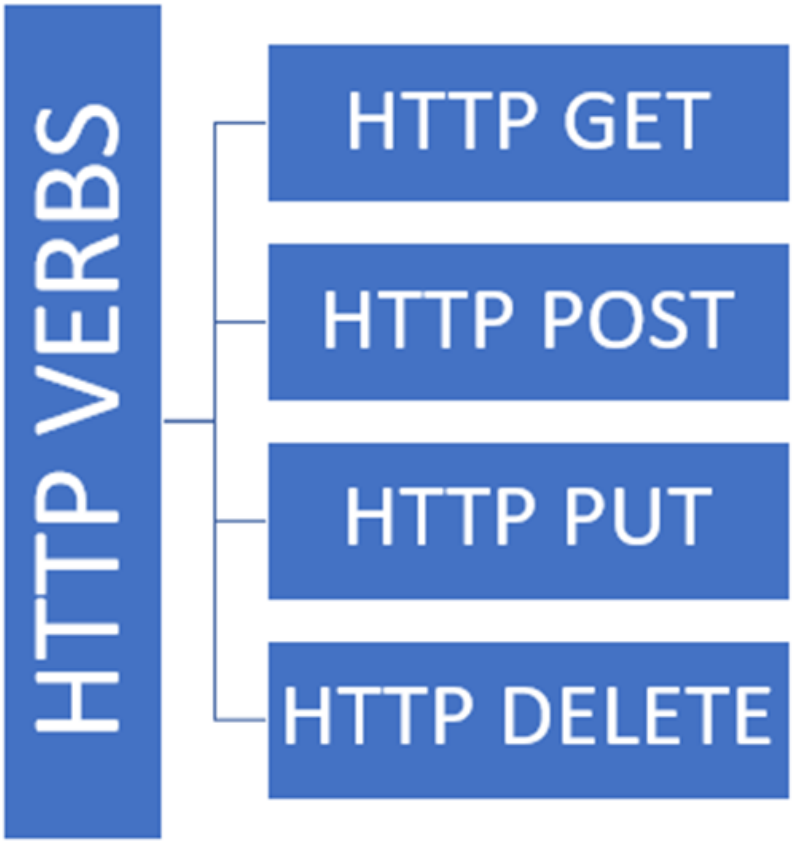

Figure 36 Http Operations

- HTTP GET – Read Operation.
- HTTP POST – Create Operation.
- HTTP PUT – Update Operation.
- HTTP DELETE – Delete Operation.

7.6. API consists of

7.6.1. Models:

The model classes represent domain-specific data and business logic in the MVC application. It represents the shape of the data as public properties and business logic as methods.

In the ASP.NET MVC API, all the Model classes must be created in the Model folder.

The model class represents a table in the database of the system.

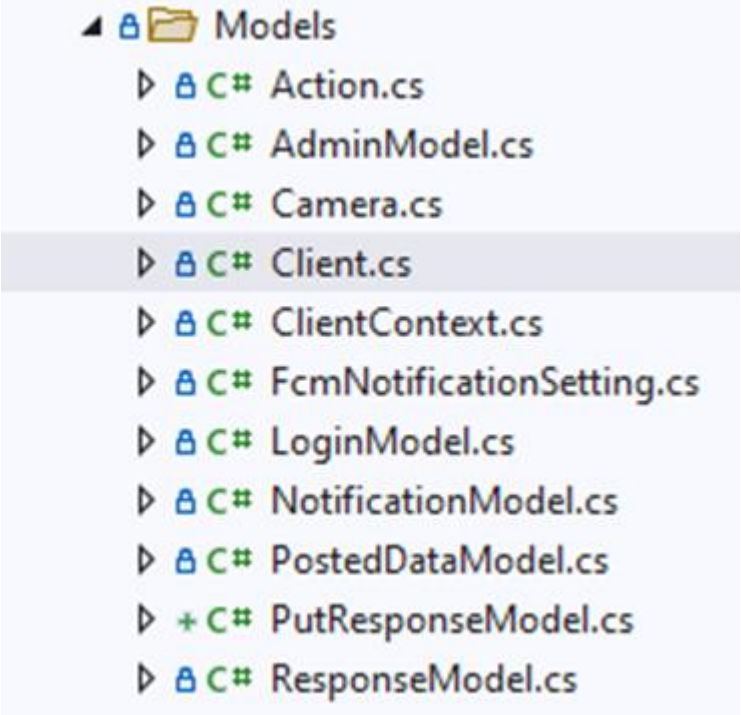

Figure 37 API Model Class

Example of the structure of the models: (Posted Data Model)

```
17 references
public class PostedDataModel
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    [Display(Name = "Posted Data Id")]
    2 references
    public int PostedDataId { get; set; }

    [Required]
    [Column(TypeName = "varchar(300)")]
    [Display(Name = "Anomaly Screenshot")]
    5 references
    public string? CrimeScreenshot { get; set; }

    [Required]
    [Column(TypeName = "varchar(150)")]
    [Display(Name = "Anomaly Date and Time")]
    5 references
    public string? AnomalyDateTime { get; set; }

    [Required]
    [Column(TypeName = "varchar(150)")]
    [Display(Name = "Anomaly type")]
    [MaxLength(100)]
    8 references
    public string? AnomalyType { get; set; }

    [Required]
    [Column(TypeName = "varchar(15)")]
    [Display(Name = "Action Priority")]
    [MaxLength(10)]
    5 references
    public string? ActionPriority { get; set; }

    [ForeignKey("Camera")]
    [Display(Name = "Zone ID")]
    [NotMapped]
    6 references
    public int ZoneID { get; set; }

    1 reference
    public string respone { get; set; }
}
```

Figure 38 API Posted Data Model Class

7.6.2. Controllers:

Web API introduces a new route for API calls as well as an APIController that responds to Get, Post, Put, and Delete requests from REST Clients. Web API introduces formatters that can do content negotiation via XML, JSON, etc. as well as provides better customization of the response using HttpResponseMessage.

Each model has its own controller to access and manipulate its members.

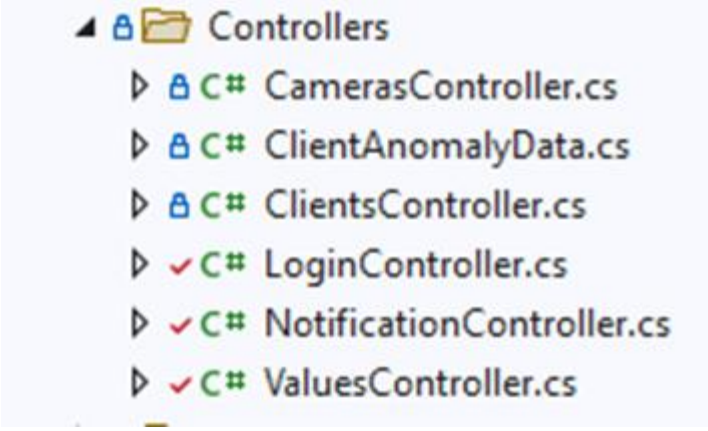

Figure 39 API Controllers

Example of the structure of the controllers: (HTTP Post request for Posted Data controller)

```
0 references
public async Task<string> Post([FromForm] string value, [FromForm] string dt, [FromForm] string zone, [FromForm] string anomalyType
{
        //Anomaly data recive part:

    PostedDataModel pdm = new PostedDataModel();
    pdm.CrimeScreenshot = value;
    pdm.AnomalyDateTime = dt;
    pdm.ZoneID = Int32.Parse(zone);
    pdm.AnomalyType = anomalyType;
    pdm.ActionPriority = anomalyPriority;
    string result = PostValuesDB.PostValuesToDB(pdm);

        //send notification part:

    NotificationModel nm = Data.SendNotification.GetDeviceIDFromDB(Int32.Parse(zone), anomalyType);
    ResponseModel notificationResult = await _notificationService.SendNotification(nm);
    result += "\n IsSuccess : " + notificationResult.IsSuccess.ToString();

        //send action part:

    int anomalyTypeNumber;
    if (pdm.AnomalyType.Contains("Explosion"))
        anomalyTypeNumber = 2;
    else if (pdm.AnomalyType.Contains("Fighting"))
        anomalyTypeNumber = 4;
    else if (pdm.AnomalyType.Contains("Shoplifting"))
        anomalyTypeNumber = 3;
    else
        anomalyTypeNumber = 1;
    SendActionToESP send = new SendActionToESP();
    FirebaseResponse response = await send.SendActionTypeToESP(anomalyTypeNumber, pdm.ZoneID);

    return result;
```

Figure 40 API HTTP Post request for Posted Data Controller

8. Module: Database

SQL Server is a relational database management system, or RDBMS, developed and marketed by Microsoft.

Similar to other RDBMS software, SQL Server is built on top of SQL, a standard programming language for interacting with relational databases. SQL Server is tied to Transact-SQL, or T-SQL, the Microsoft's implementation of SQL that adds a set of proprietary programming constructs.

SQL Server consists of two main components:

- **Database Engine.**
- **SQLOS.**

8.1. Database Engine

The core component of the SQL Server is the Database Engine. The Database Engine consists of a relational engine that processes queries and a storage engine that manages database files, pages, indexes, etc. The database objects such as stored procedures, views, and triggers are also created and executed by the Database Engine.

8.1.1. Relational Engine

The Relational Engine contains the components that determine the best way to execute a query. The relational engine is also known as the query processor.

The relational engine requests data from the storage engine based on the input query and processed results.

Some tasks of the relational engine include querying processing, memory management, thread and task management, buffer management, and distributed query processing.

8.1.2. Storage Engine

The storage engine is in charge of storage and retrieval of data from the storage systems such as disks and SAN.

8.2. SQLOS

Under the relational engine and storage engine is the SQL Server Operating System or SQLOS.

SQLOS provides many operating system services such as memory and I/O management. Other services include exception handling and synchronization services.

8.2.1. SQL Server Services and Tools

Microsoft provides both data management and business intelligence (BI) tools and services together with SQL Server.

For data management, SQL Server includes SQL Server Integration Services (SSIS), SQL Server Data Quality Services, and SQL Server Master Data Services. To develop databases, SQL Server provides SQL Server Data tools; and to manage, deploy, and monitor databases SQL Server has SQL Server Management Studio (SSMS).

For data analysis, SQL Server offers SQL Server Analysis Services (SSAS). SQL Server Reporting Services (SSRS) provides reports and visualization of data. The Machine Learning Services technology appeared first in SQL Server 2016 which was renamed from the R Services.

8.3. SQL Server Editions

SQL Server has four primary editions that have different bundled services and tools. Two editions are available free of charge:

SQL Server Developer edition for use in database development and testing.

SQL Server Expression for small databases with the size of up to 10 GB of disk storage capacity.

For larger and more critical applications, SQL Server offers the Enterprise edition that includes all SQL Server's features.

SQL Server Standard Edition has partial feature sets of the Enterprise Edition and limits on the Server regarding the numbers of processor core and memory that can be configured.

8.4. ER Diagram

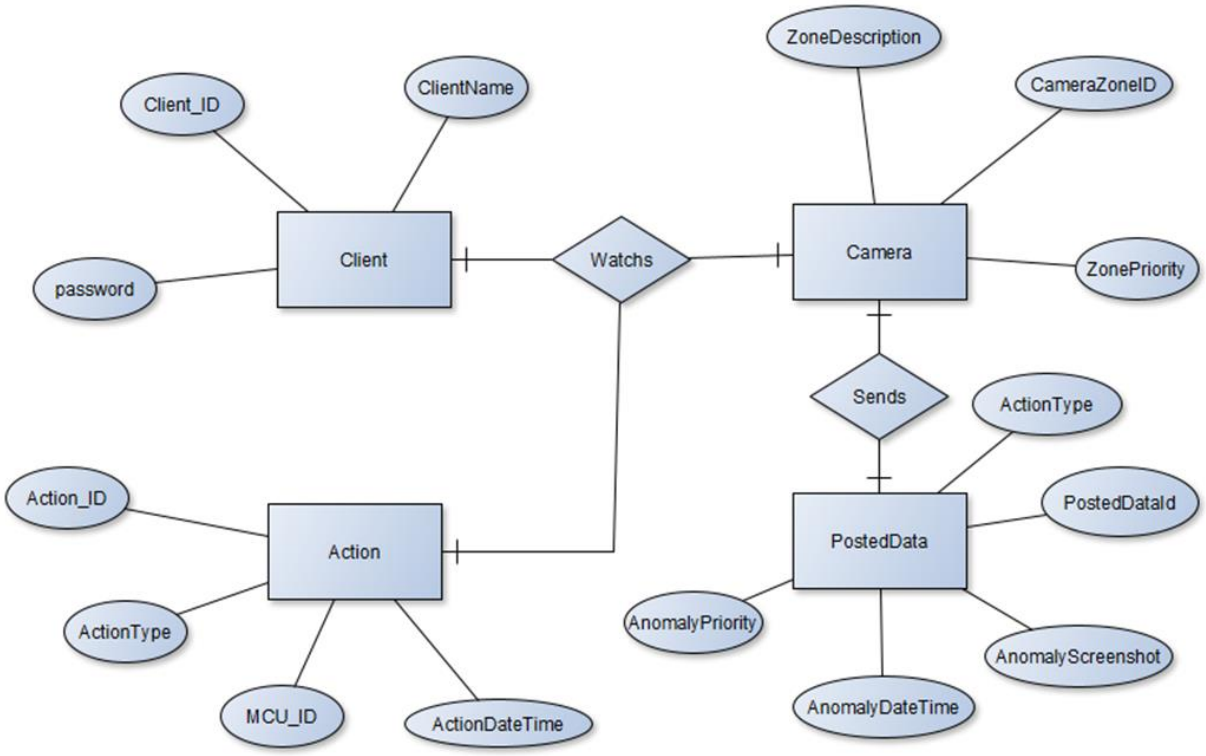

Figure 41 ERD

8.5. System tables:

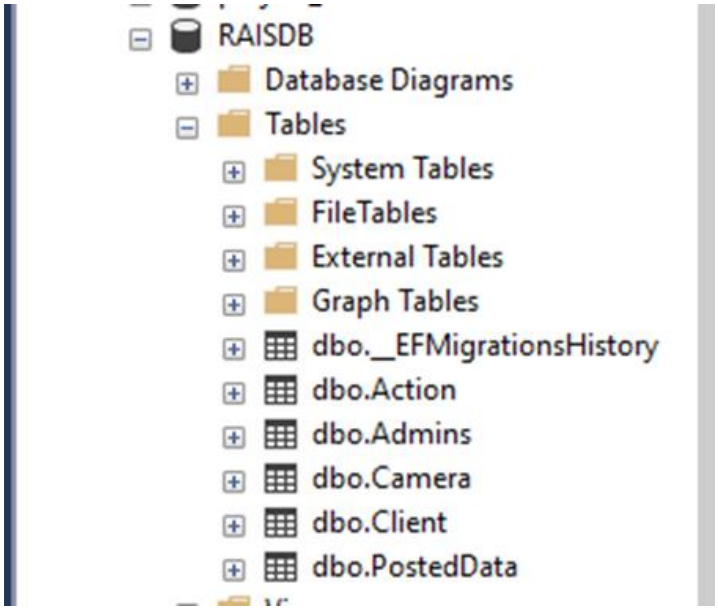

Figure 42 System Database Tables

8.5.1. Client:

Used to store the information of the security guards (clients) as:

ID, username, password, the corresponding zone, and device token which is used to send the notification to client's cell phone.

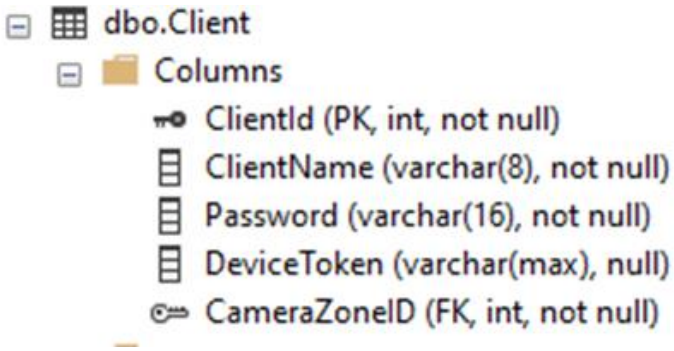

Figure 43 Database Client Table

8.5.2. Action:

Used to store the information of the hardware actions as:

ID, action type, ID of the hardware correspond to the action, and the responsible client of this action.

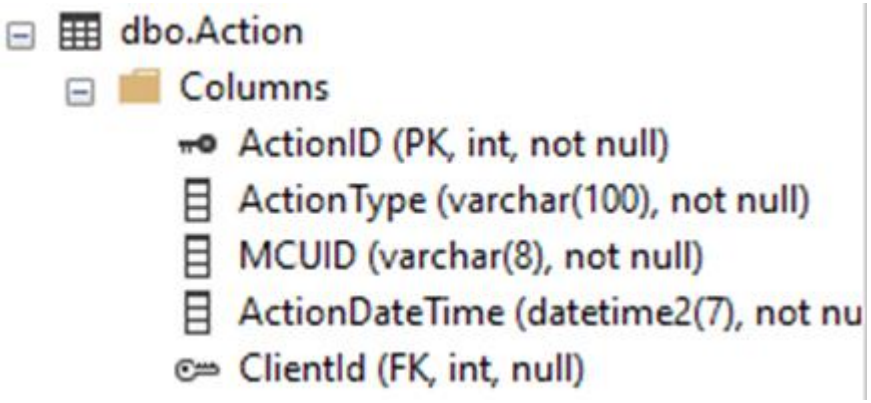

Figure 44 Database Action Table

8.5.3. Camera:

Used to store the information of the camera zones inbound the place that the system placed in as:

ID, zone priority which is used to differentiate zones according to the priority from highest to lowest.

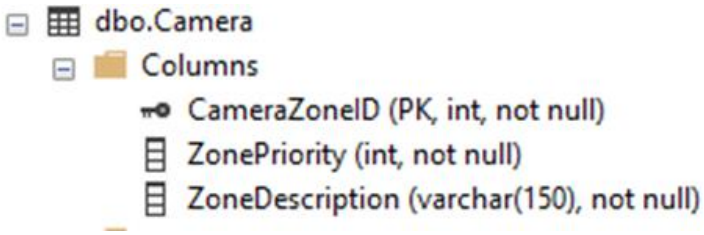

Figure 45 Database Camera Table

8.5.4. Admins:

Used to store the information of the admin of the system who will use the website to monitor the whole process.

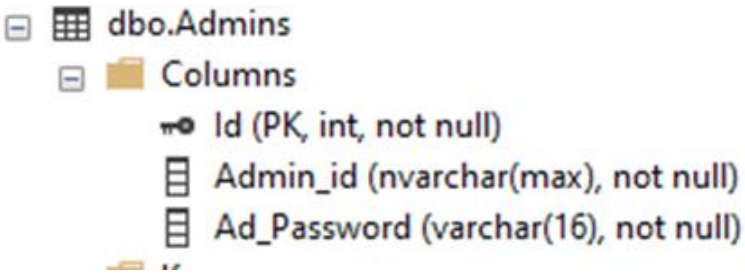

Figure 46 Database Admins Table

8.5.5. Posted Data:

Used to store the information of anomaly events.

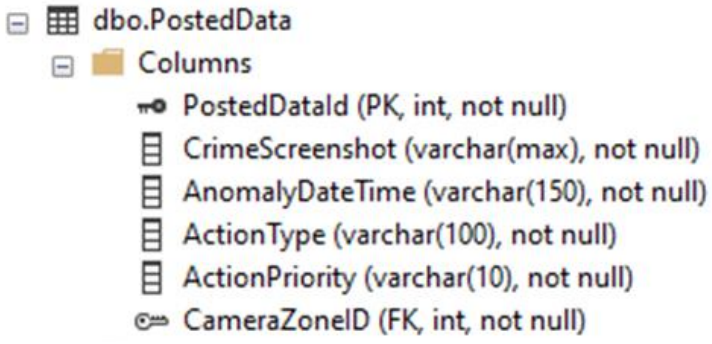

Figure 47Database Posted Data Table

9. Module: Mobile Application

The mobile application is used by the security guards (clients) to receive the anomaly event information as: snapshot of the anomaly event, the date and time, the anomaly event type, and the zone that the anomaly event took place in.

The application consisting of 2 Activities:

9.1. Login Activity:

Figure 48 Mobile App. Login Screen

Used for authentication and authorization of the clients.

The admin of the system has the ability of add, edit, and delete clients as we will see in the admin's application.

Once the client logged in for the first time the device token which used to specify a device for notification sending process will be inserted to the database automatically in the device token column.

This insertion process will be useful if a client gets another mobile once he logged in the token will be updated automatically without need the admin to update it manually.

Remember me, the feature is useful to keep the client logged in without needing to log in every time he opens the application.

9.2. Main Activity:

Figure 49Mobile App. Main Screen

This activity shows the snapshot, date and time, type, and the zone of the anomaly event.

There's a timeout of the screenshot, after that it will be updated each 10 seconds if a new event occurred.

The notification will be as following fig.

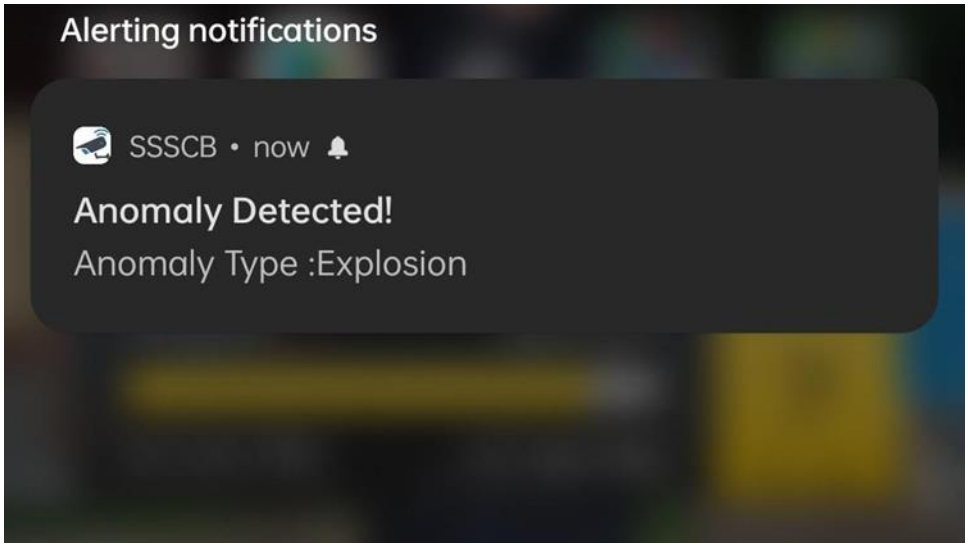

Figure 50 Mobile App. Notification

Notification sending process is handled by our API with these steps:

- Whenever an anomaly event occurred the ML model will send the screenshot, date and time, zone, and anomaly type to Posted data Controller of the API.
- According to the zone the image will be inserted into the database.
- Also, according to the zone a query will be generated to retrieve device token using zone id to determine which client is related to this zone to send notification to.
- Using device token the notification will be pushed by Google Firebase Cloud Messaging (FCM).

10. Module: Admin Interface

The admin of the system can monitor and handle all of these operations:

- Different camera streams.
- Add, edit, delete clients.
- Add, edit, delete zones.
- Show full history of anomaly events.
- Show full history of hardware actions.

This interface is built with Angular 13.

10.1. Camera streams monitoring:

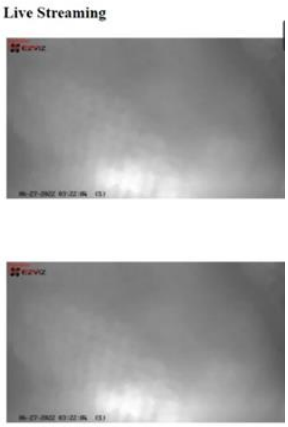

Figure 51 Stream Monitoring

10.2. Monitoring clients:

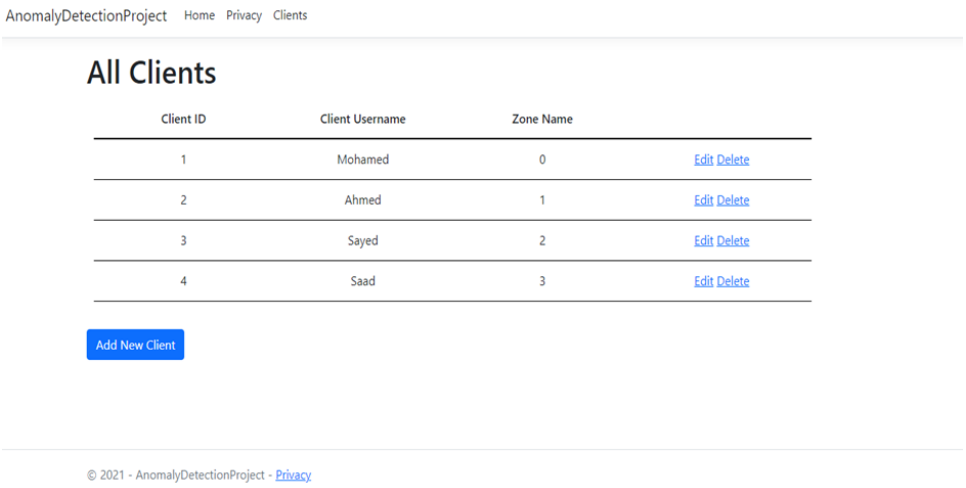

Figure 52 Admin Monitoring Clients Page

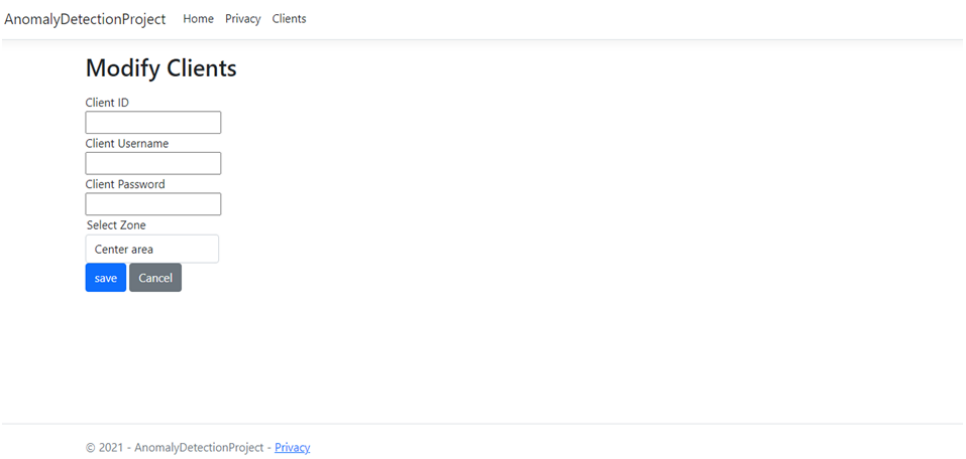

Figure 53 Admin Edit Client Page

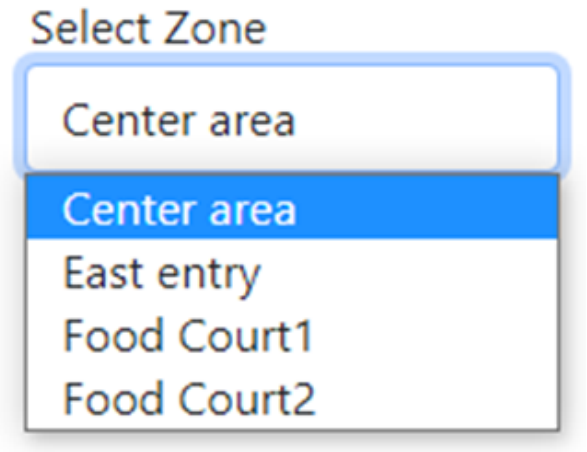

Figure 54 Selecting Zone for Client

10.3. Viewing Full Actions History:

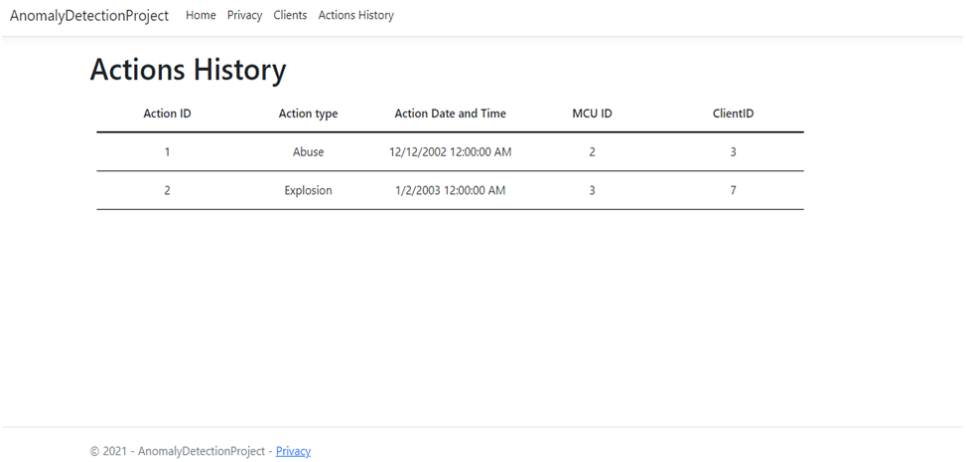

Figure 55 Admin Action History Page

10.4. Monitoring Zones:

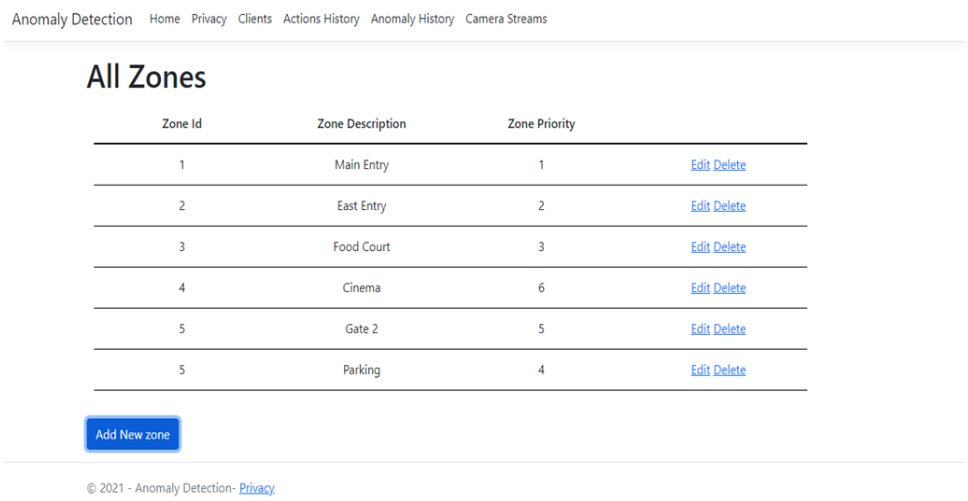

Figure 56 Admin Zone Edit Page

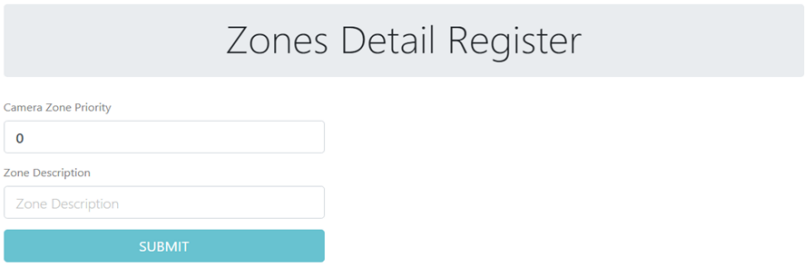

Figure 57 Admin Adding New Zone Page

11. Module: Stream of IP Cameras

The C6N from EZVIZ comes equipped with a Smart IR function, which uses advanced infrared (IR) lighting to capture more details in dim light. With its 360-degree field of view and smart tracking function, you won't need to worry about missing a thing.

Figure 58 EZVIZ C6N IP Camera

Camera features:

- Easy API for frames streaming.
- Low Cost.
- IP camera, so, we can connect it through Wi-Fi and NVR in case of multiple cameras.

11.1. IP Camera Streaming Using RTSP Protocol:

The Real Time Streaming Protocol (RTSP) is an application-level network protocol designed for multiplexing and packetizing multimedia transport streams (such as interactive media, video and audio) over a suitable transport protocol.

We have used **port forwarding** to access the RTSP stream from the website.

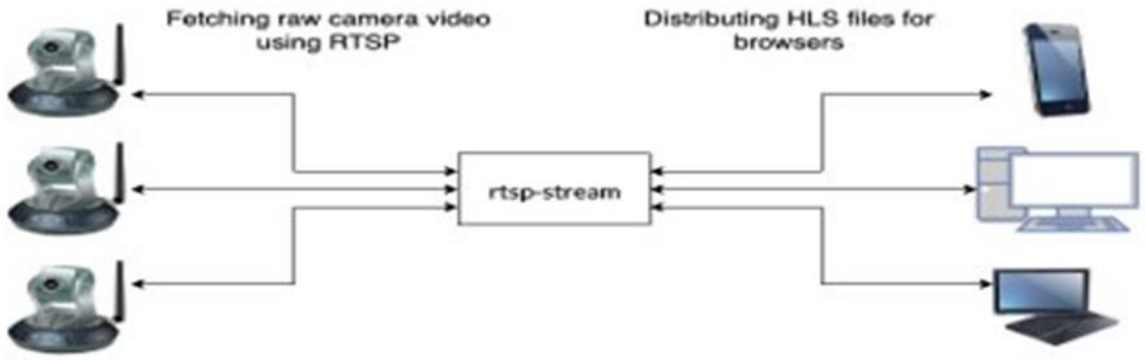

Figure 59 RSTP Block diagram

Port forwarding or **port mapping** is an application of network address translation (NAT) that redirects a communication request from one address and port number combination to another while the packets are traversing a network gateway, such as a router or firewall.

This technique is most commonly used to make services on a host residing on a protected or masqueraded (internal) network available to hosts on the opposite side of the gateway (external network), by remapping the destination IP address and port number of the communication to an internal host.

Port forwarding allows remote computers (for example, computers on the Internet) to connect to a specific computer or service within a private local-area network (LAN).

When configuring port forwarding, the network administrator sets aside one port number on the gateway for the exclusive use of communicating with a service in the private network, located on a specific host. External hosts must know this port number and the address of the gateway to communicate with the network-internal service. Often, the port numbers of well-known Internet services, such as port number 80 for web services (HTTP), are used in port forwarding, so that common Internet services may be implemented on hosts within private networks.

Alos for testing purposes we have used Webcams and Mobile cams.

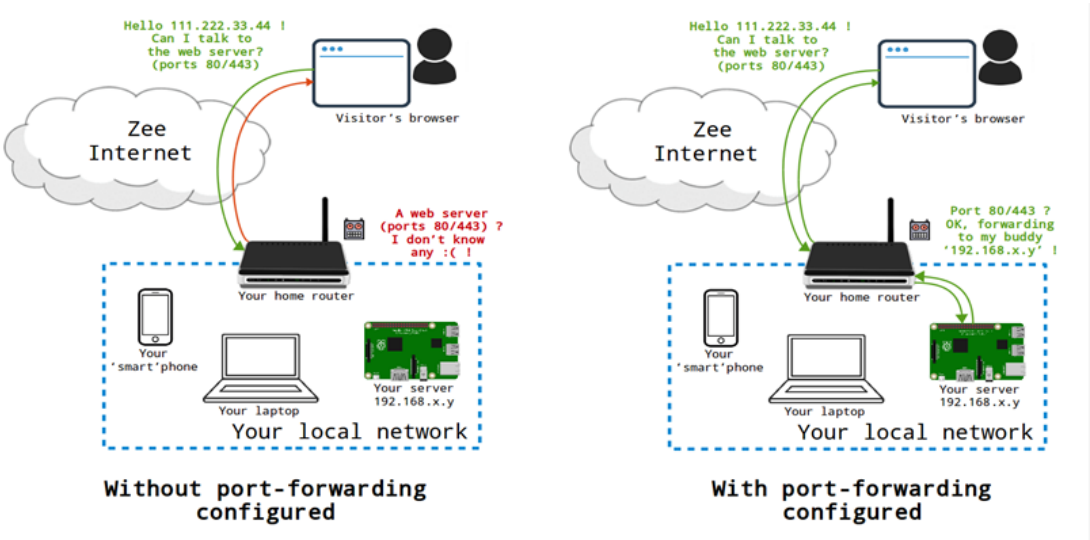

Figure 60 Port Forwarding

12. Module: Local hosting on IIS Express

IIS Express is a lightweight, self-contained version of IIS optimized for developers. IIS Express makes it easy to use the most current version of IIS to develop and test websites. It has all the core capabilities of IIS 7 and above as well as additional features designed to ease website development including:

- It doesn't run as a service or require administrator user rights to perform most tasks.
- IIS Express works well with ASP.NET and PHP applications.
- Multiple users of IIS Express can work independently on the same computer.

Enable IIS from Windows features as following:

- go to start.

- type windows features.

- look for Internet Information Services.

- enable FTP server: FTP service.

- enable Web Management Tool: IIS Management Console.

- enable all services of World Wide Web Services.

Then open IIS from:

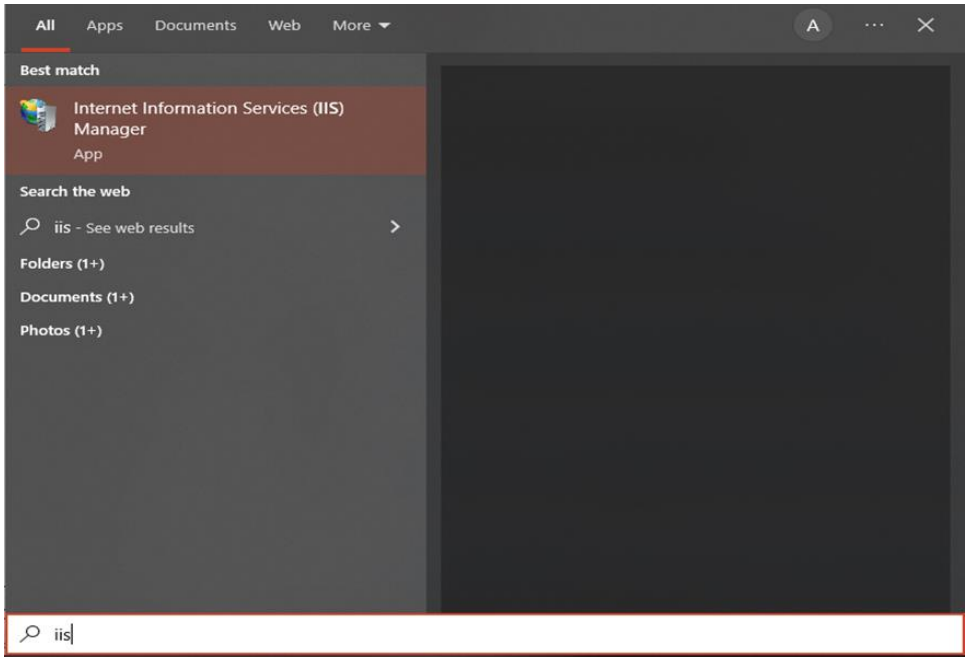

Figure 61 ISS Open Icon

IIS Interface:

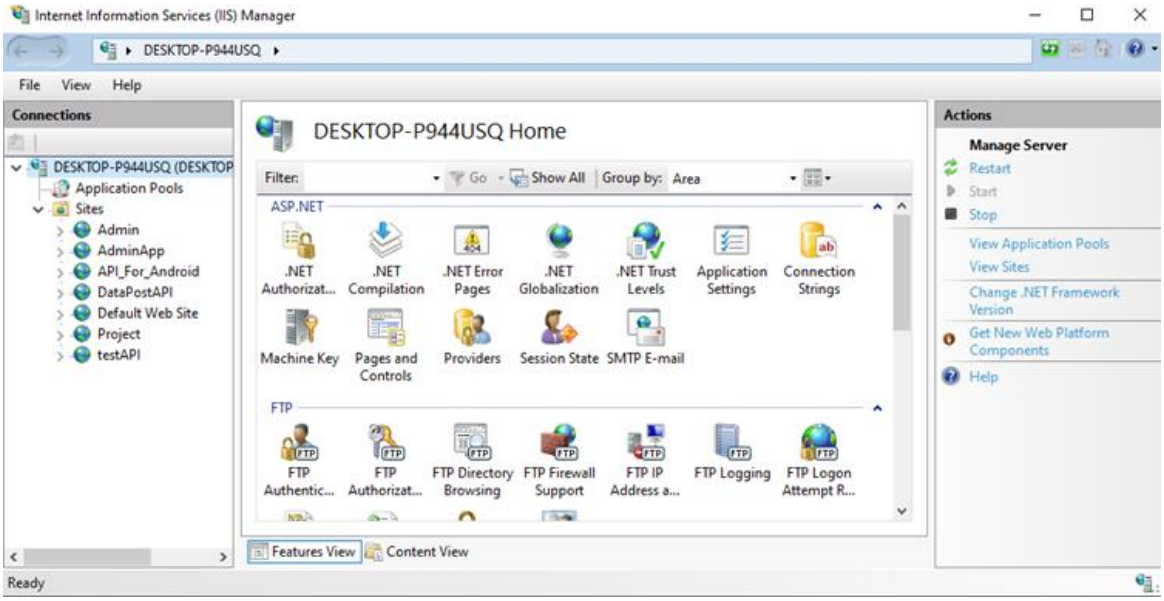

Figure 62 ISS Interface

Also, for testing purposes we have used Conveyor.

Conveyor Allows you to access your web applications from other machines such as iOS and Android devices, even if the web application is hosted on the development web server. Use this when testing your web project from desktop computers, phones or tablets.

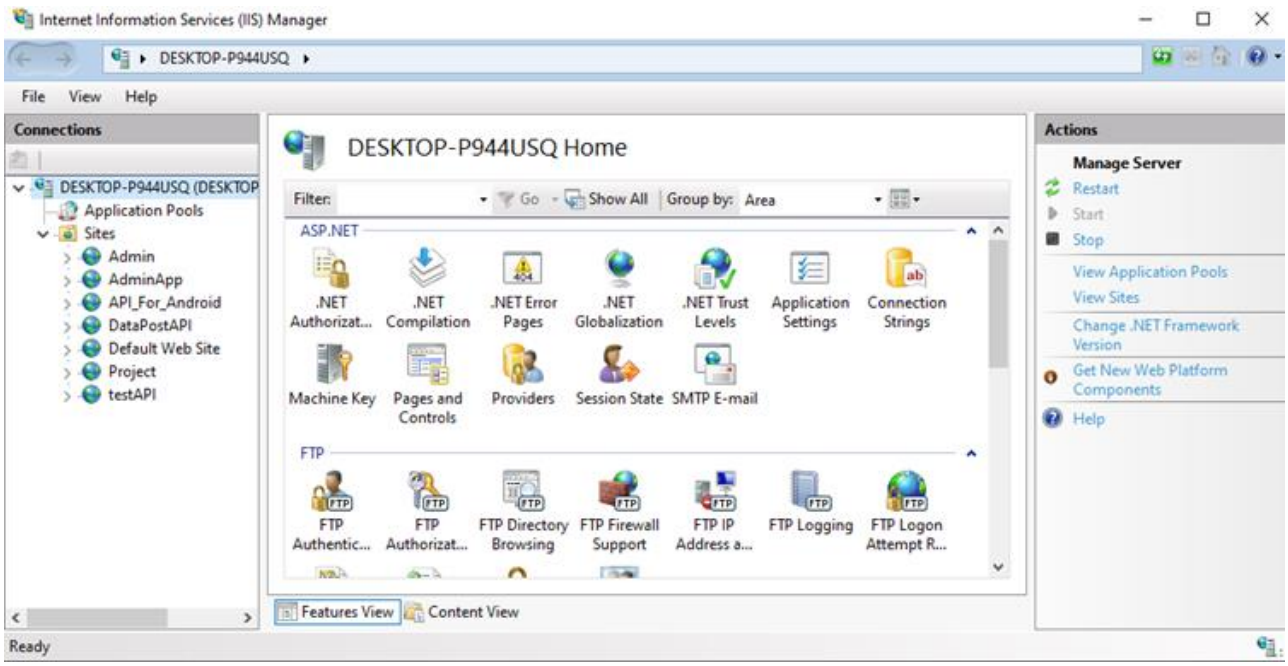

Figure 63 IIS Manager.

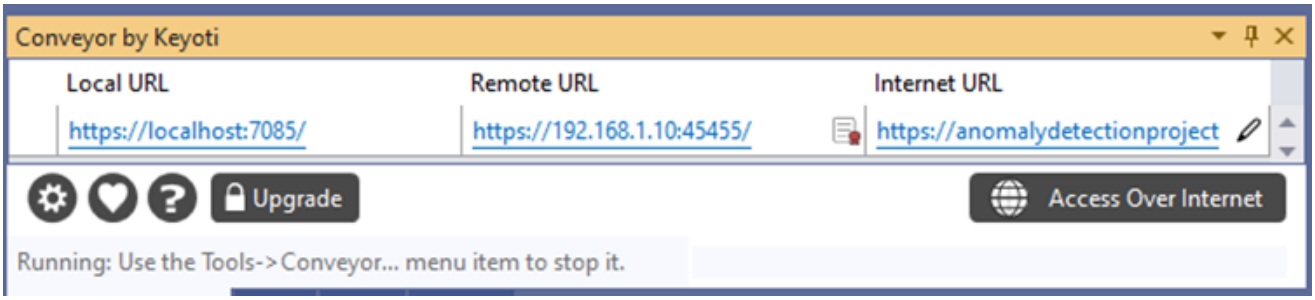

Figure 64 Conveyor GUI.

13. Firebase DB for Notifications and Actions

Firebase Cloud Messaging (FCM) is a set of tools that sends push notifications and small messages of up to 4 KB to different platforms: Android, iOS and web.

Firebase Cloud Messaging has a simple architecture with four main parts:

- A service, API or console that sends messages to targeted devices.
- The Firebase Cloud Messaging back end, where all the processing happens.
- A transport layer that's specific to each platform. In Android's case, this is called the Android Transport Layer.
- The SDK on the device where you'll receive the messages. In this case, called the Android Firebase Cloud Messaging SDK.

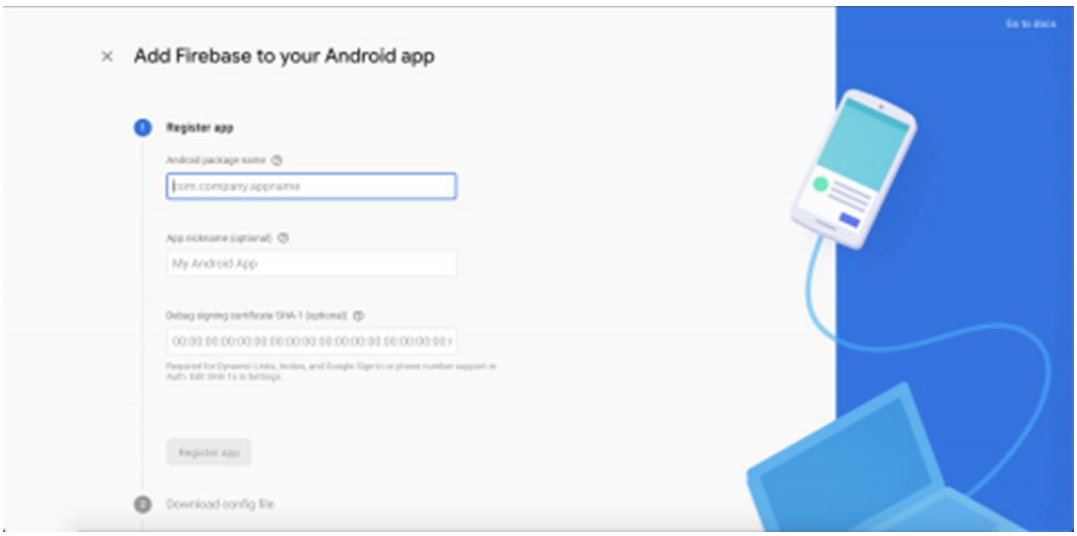

Figure 65 Firebase GUI.

Before receiving any push notifications in your app, there's a series of steps you'll need to configure in the Firebase console.

To start integrating Firebase with your project, click the **Android** button, which is under **Get started by adding Firebase to your app**. You'll see a screen requesting data from your app.

6.1. Using the Device Token:

The **device token** is a unique identifier that contains two things:

- Which device will receive the notification.
- The app within that device that will receive the notification.

Firebase automatically generates these tokens and delivers them to you as a developer.

6.2. Retrieving the Device Token:

Device tokens may change over time, due to circumstances like:

- The app deleted the Instance ID.
- The user got a new device and restored the app.
- The user uninstalled and reinstalled the app.
- The user cleared app data.

In these cases, you'll need to retrieve the device token.

In most of your apps, you'll retrieve this token in the background. Sometimes, you'll need to save it to a database so you can use it to send push notifications with internal APIs. In this app, you already have a **Retrieve token** button, which you'll use so you can see the operation's result.

Here's what this code does, step-by-step:

- You get the instance of the app that's tied to the Firebase back end and add a complete listener to it so you know when the task finishes, whether with an error or success.
- You check if the task, which is the result of this function, isn't successful and return an error to the console.
- If the task is successful, you get the result from it.
- After getting the token, you set it as a string and output it for database using PUT request.

6.3. For Action sending to ESP 8266:

Create Fire store database on Firebase console.

This database has 2 columns: Anomaly type, and Zone.

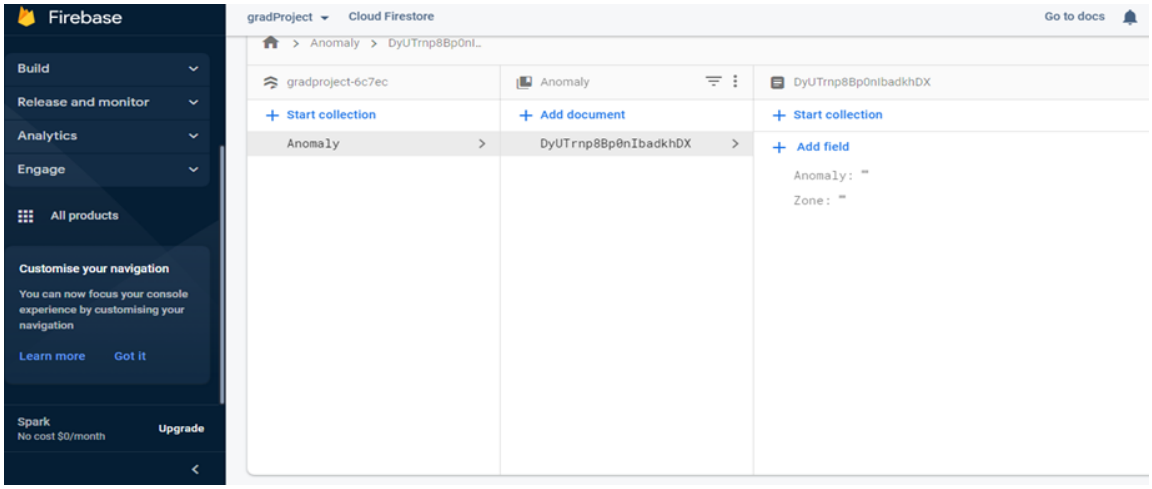

Figure 66 fire store creation from firebase.

Then using REST API, we can write and update Firebase Realtime Database Rules for your Firebase app by making a `PUT` request to the `/.settings/rules.json` path. To do this, we'll need an access token to authenticate our REST request.

So, we will use HTTP PUT request to modify values of Anomaly type, and Zone according to the corresponding action and zone for each anomaly event.

```
4 references
public class SendActionToESP
{
    2 references
    public async Task<FirebaseResponse> SendActionTypeToESP(int id,int zoneNumber)
    {
        FirebaseDB firebaseDB = new FirebaseDB("https://esp8266-27419-default-rtdb.firebaseio.com/");

        // Referring to Node with name "Teams"
        FirebaseDB firebaseDBTeams = firebaseDB.Node("anomaly"+zoneNumber.ToString());

        var data = "{"+  "int"+": "+id + "}";

        FirebaseResponse postResponse = await firebaseDBTeams.Put(data);
        return postResponse;

    }
}
```

Figure 67 API Send Action to ESP Class

14. Use Case Diagram

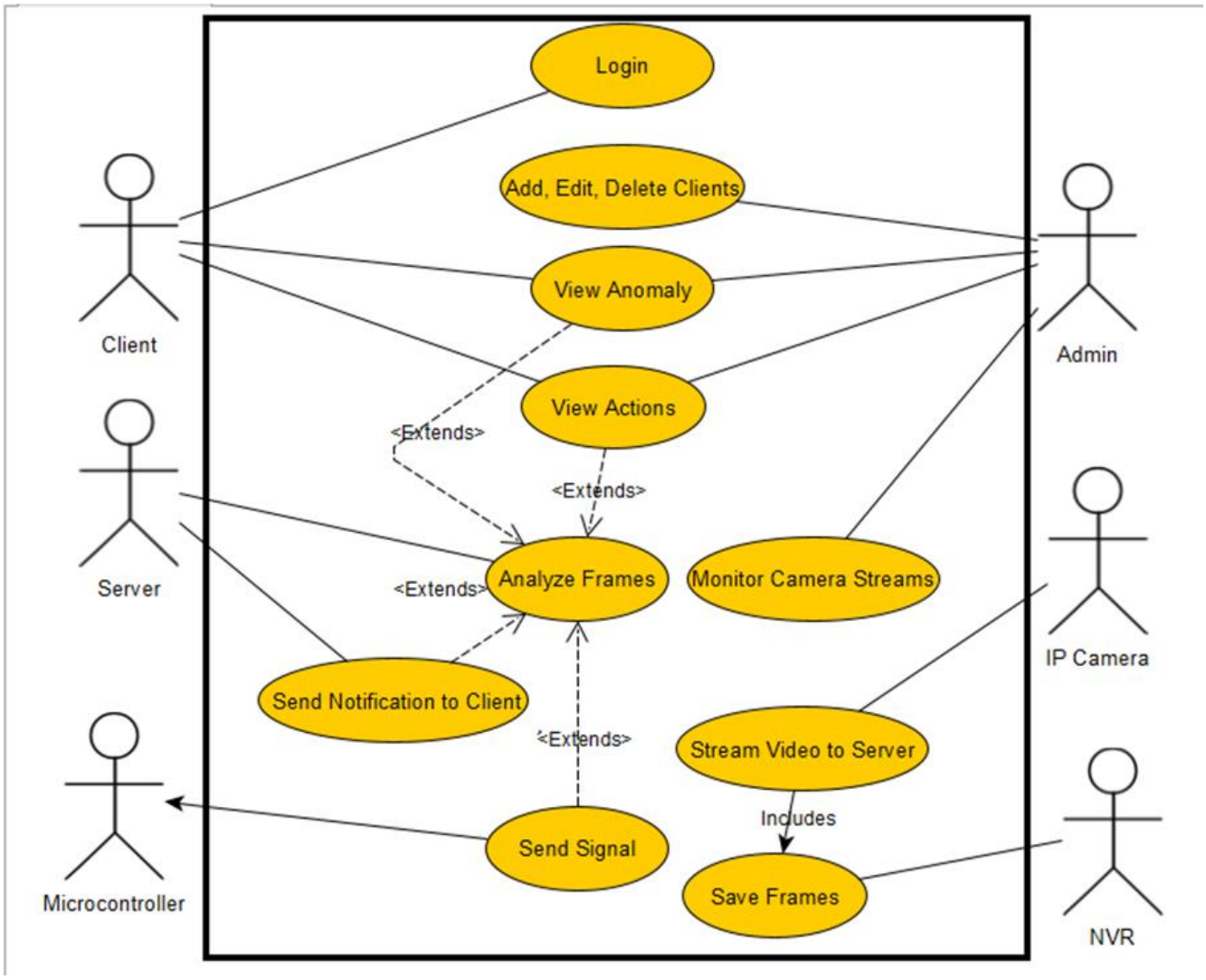

Figure 68 Use Case Diagram

15. Sequence Diagrams

15.1. Login Sequence:

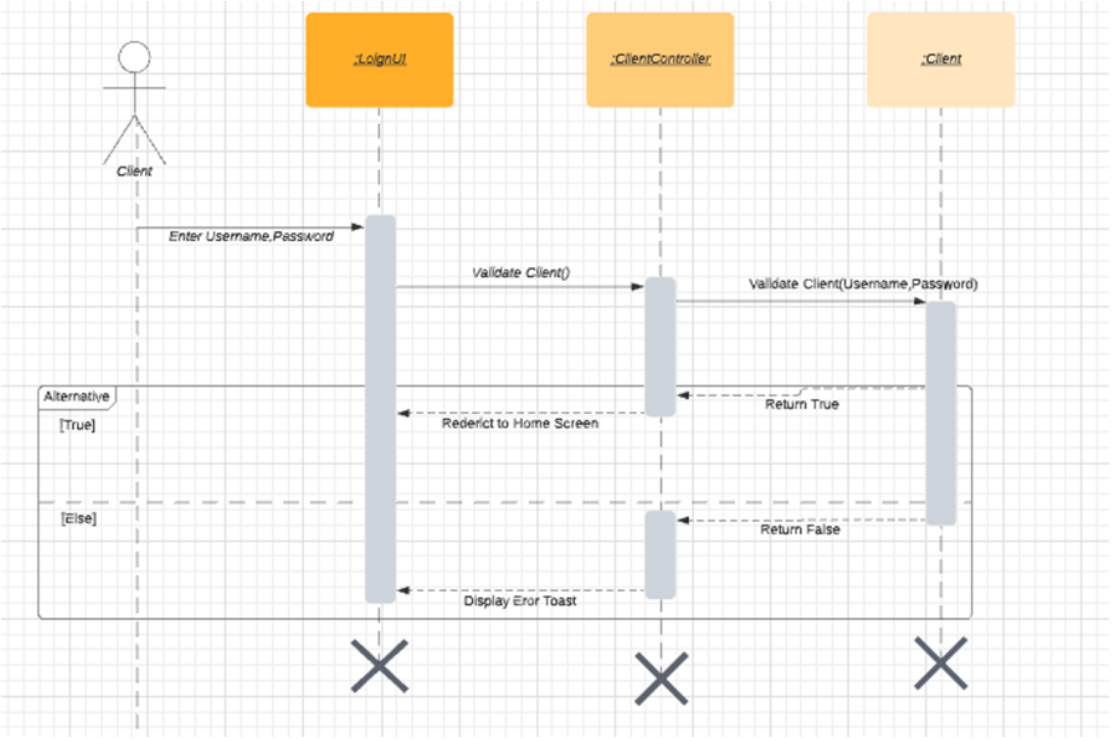

Figure 69 Login Sequence Diagram

15.2. Add, edit, and delete sequence:

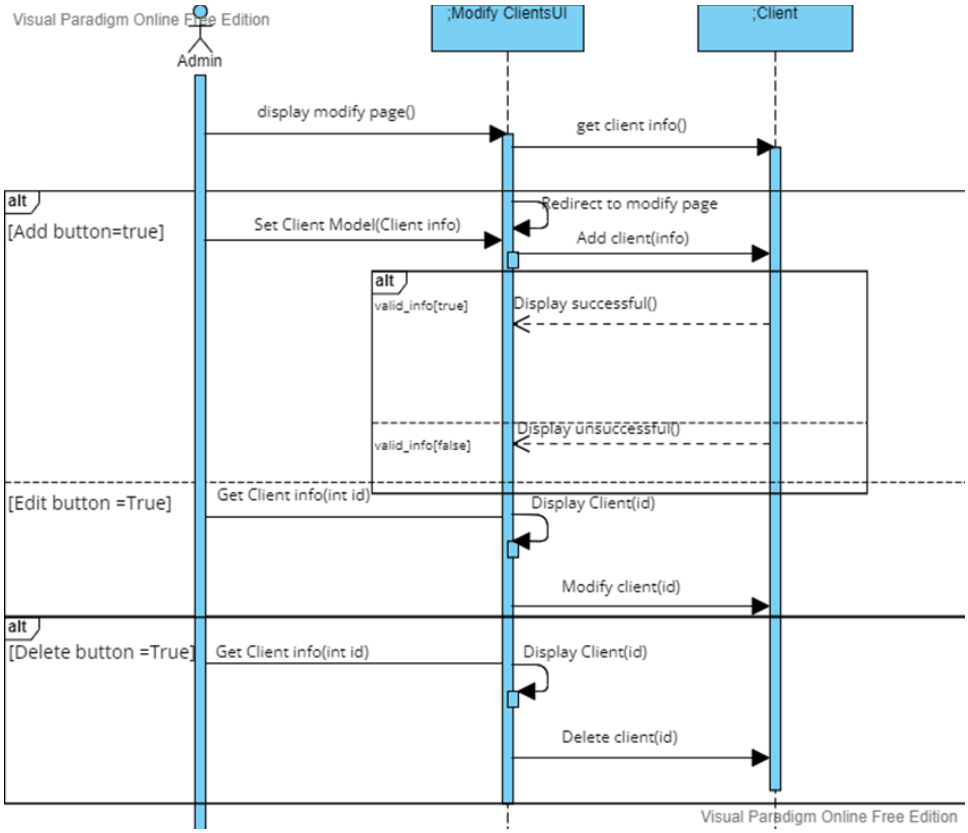

Figure 70 Add, Edit, Delete Sequence Diagram

15.3. View anomaly information sequence:

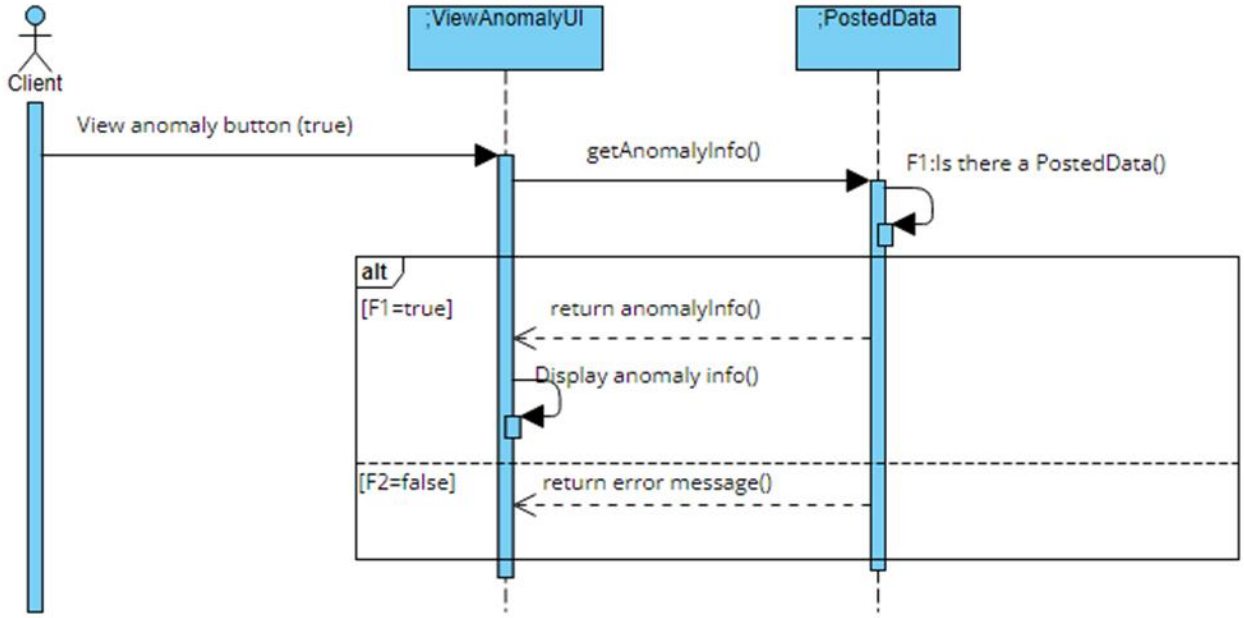

Figure 71 View Anomaly Information Sequence Diagram

15.4. View anomaly history sequence:

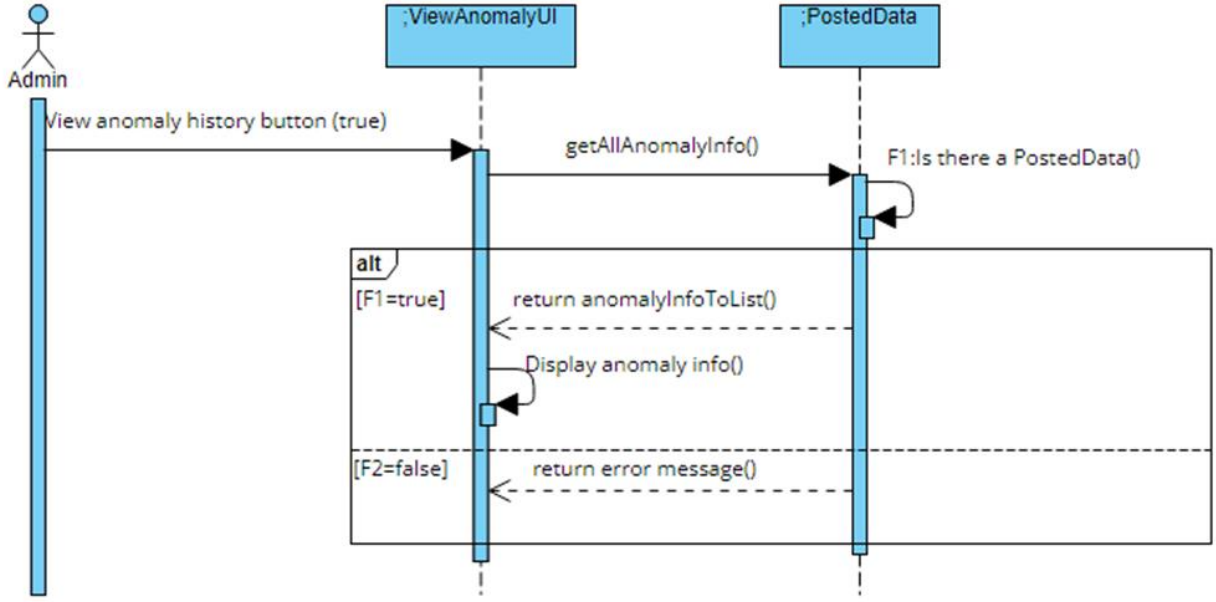

Figure 72 View Anomaly History Sequence Diagram

16. Hardware Action

We wanted to make the system smarter and faster by dealing with situations so we used esp8266 MCU to communicate with the system and to take quick actions. We have connected every situation according to its type to a specific action that fits the situation. In our system if a fight took place, an alarm would be turned on and a message would be sent to the mobile application to inform the security and the owner of the situation. In our system if an explosion takes place, an alarm will be turned on and a message will be sent to the mobile application to inform the security and the owner of the situation.

16.1. Component we have used for hardware actions

- **ESP8266MCU**
- **LEDS**
- **Buzzers**
- **Wires**
- **Battery**
- **Hotspot**
- **Realtime database**
- **Telegram bot**
- **Arduino IDE**

16.2. Esp8266

Node MCU is a low-cost open source IoT platform.

It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espress if Systems, and hardware which was based on the ESP-12 module.

16.2.1. Specifications of esp8266

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106.
- Operating Voltage: 3.3V.
- Input Voltage: 7-12V.
- Digital I/O Pins (DIO): 16.
- Analog Input Pins (ADC): 1.
- UARTs: 1.
- SPIs: 1.
- I2Cs: 1.
- Flash Memory: 4 MB.
- SRAM: 64 KB.
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug in Play.
- PCB Antenna.
- Small Sized module to fit smartly inside your IoT projects.

16.2.2. Pinout of esp8266

Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	**EN, RST**	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

Figure 73 ESP 8266 Pinout

16.2.3. ESP Block Diagram

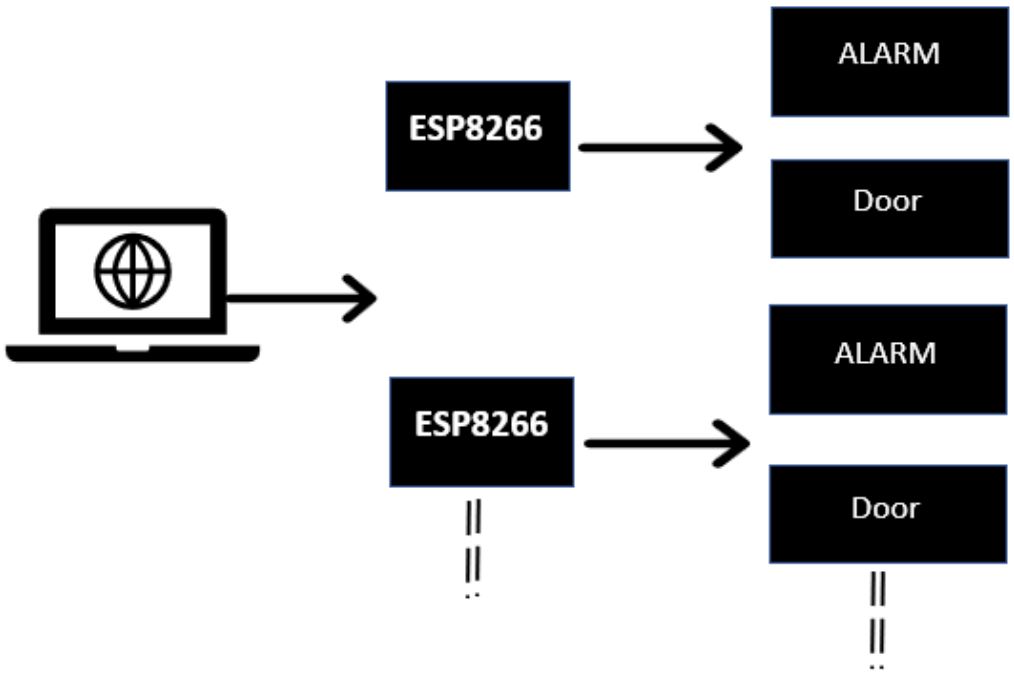

Figure 74 Hardware Action Block Diagram.

16.3. Steps

16.3.1. Creating a Telegram Bot

Get Token:

- Install telegram.
- Open Telegram and First, search for "botfather".
- Click the start button.
- Type /newbot and follow the instructions to create your bot. Give it a name and username.
- If your bot is successfully created, you'll receive a message with a link to access the bot and the bot token. Save the bot token because you'll need it so that the ESP8266 can interact with the bot.

Get Your Telegram User ID:

- In your Telegram account, search for "IDBot"**.**
- Start a conversation with that bot and type **/getid**. You will get a reply back with your user ID. Save that **user ID**

Install libraries:

Universal Telegram Bot Library:

To interact with the Telegram bot, we'll use the Universal Telegram Bot Library created by Brian Lough that provides an easy interface for the Telegram Bot API.

Follow the next steps to install the latest release of the library:

- Download the Universal Arduino Telegram Bot library.
- Go to Sketch > Include Library > Add.ZIP Library.
- Add the library you've just downloaded.

Arduino-Json Library.

You also have to install the ArduinoJson library. Follow the next steps to install the library.

- Go to Sketch > Include Library > Manage Libraries.
- Search for "ArduinoJson".
- Install the library.

We're using ArduinoJson library version 6.5.12.

16.3.2. Set Up a Firebase

Set up a new fire base for ESP 8266.

- Create a New Project.
- Set Authentication Methods.
- Create a Realtime Database.
- Get Project API Key.
- Program the ESP8266 to Interface with Firebase.

To program the ESP8266, you can use Arduino IDE, VS Code with the PlatformIO extension, or other suitable software.

Installation for – Arduino IDE.

- Go to Sketch > Include Library > Manage Libraries.
- Search for Firebase ESP Client and install the Firebase Arduino Client Library for ESP8266 and ESP32 by Mobitz.

16.3.3. Build Circuit.

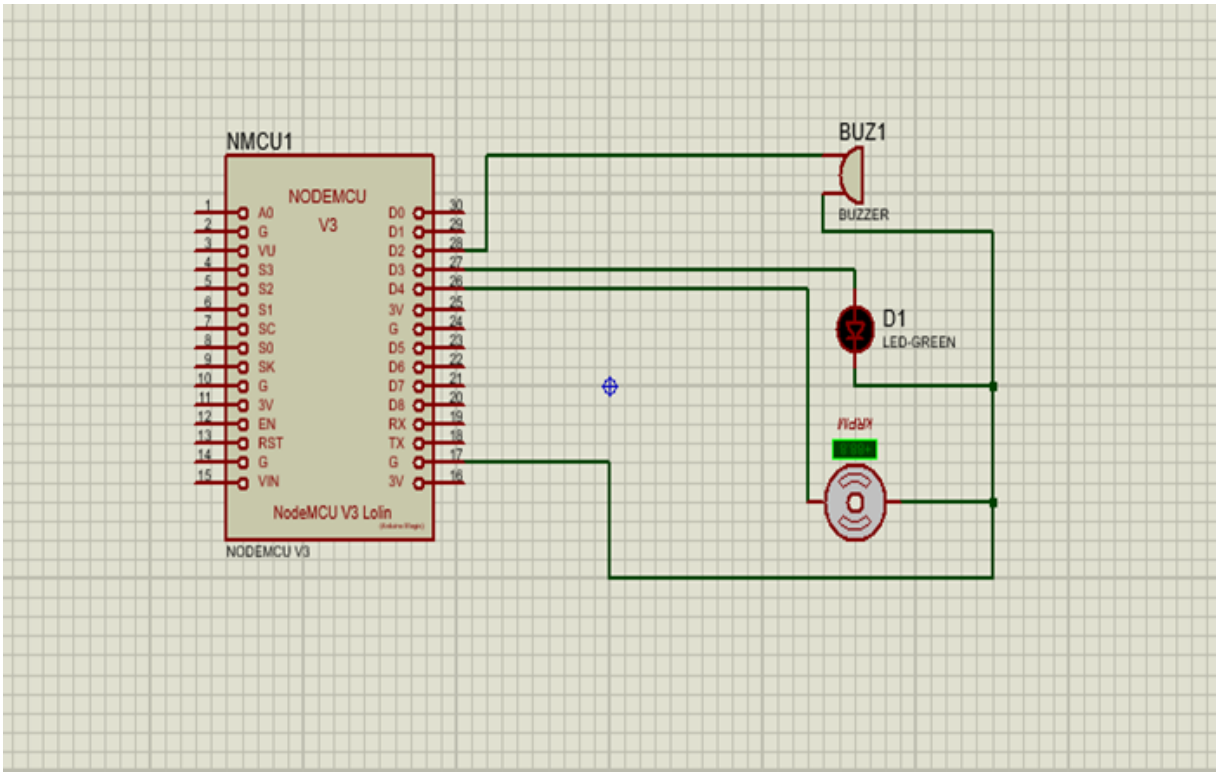

Figure 75 Hardware Action Circuit

16.3.4. Upload the code after adding all required details

Include your network credentials in the following lines.

```
#define WIFI_SSID "REPLACE_WITH_YOUR_SSID"
#define WIFI_PASSWORD "REPLACE_WITH_YOUR_PASSWORD"
```

Insert your firebase project API key.

```
#define API_KEY "REPLACE_WITH_YOUR_FIREBASE_PROJECT_API_KEY"
```

Insert your database URL.

```
#defineDATABASE_URL"REPLACE_WITH_YOUR_URL".
```

Insert your Telegram Bot token you've got from Botfather on the `BOTtoken` variable.

```
#defineBOTtoken"XXXXXXX - XXXXXXXXXXXXXXXXXXXXXXXXX"
```

Insert your chat ID. The one you've got from the IDBot.

```
#define CHAT_ID "XXXXXXXXXX"
```

17. Module: Used Components

- IIS server.
- .Net 6.
- SQL server.
- Flask.
- Java.
- Android Studio.
- FCM Google Firebase.
- Python.
- OpenCV.
- Tensorflow.
- Keras.
- Google Colab.
- EZVIZ IP camera.
- NVR.
- EZViZ API.
- Arduino MCU.
- Node MCU esp8266.

18. References

1. A world with a billion cameras watching you is just around the corner.
2. Surveillance Cameras Are Everywhere. And They're Only Going to Get More Ubiquitous. ‹ CrimeReads
3. CRCV | Center for Research in Computer Vision at the University of Central Florida (ucf.edu)
4. Vijay Mahadevan, Weixin LI, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pages 1975–1981,2010.
5. Unusual crowd activity dataset f the university of Minnesota in http://mha.cs.umn.edu/movies/crowedactivity-all.avi.
6. A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-timeunusual event detection using multiple fiXED-location monitors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(3):555–560, 2008.
7. C. Lu, J. Shi, and J. Jia. Abnormal event detection at 150 fps in matlab. In 2013 IEEE International Conference on Computer Vision, pages 2720–2727, 2013.
8. Real-Time Anomaly Detection and Feature Analysis Based on Time Series for Surveillance Video [2020 5TH International Conference on Universal Village (UV) | 978-1-7281-9523-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/UV50937.2020.9426191]
9. M Naphade, M Chang, A Sharma, D C. Anastasiu, V Jagarlamudi, P Chakraborty, T Huang, S Wang, M Liu, R Chellappa, J Hwang, and S Lyu. The 2018 nvidia ai city challenge. In Proc. CVPR Workshops, pages 53—60, 2018.
10. Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey, 2019.
11. Santhosh Kelathodi Kumaran, D. P. Dogra, and P. Roy. Anomaly detection in road traffiC using visual surveillance: A survey. ArXiv, abs/1901.08292, 2019.
12. G. Sreenu and M A Durai. Intelligent video surveillance: a review through deep learning techniques for crowd analysis. Journal of Big Data, 6:48, 06 2019.

13. TOLGA ERGEN AND S. KOZAT. UNSUPERVISED ANOMALY DETECTION WITH LSTM NEURAL NETWORKS. IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, 31:3127–3141, 2020.
14. GYUWAN KIM, HAYOON YI, JANGHO LEE, Y. PAEK, AND S. YOON. LSTM-BASED SYSTEM-CALL LANGUAGE MODELING AND ROBUST ENSEMBLE METHOD FOR DESIGNING HOST-BASED INTRUSION DETECTION SYSTEMS. ARXIV,
15. RUNTIAN ZHANG AND QIAN ZOU. TIME SERIES PREDICTION AND ANOMALY DETECTION OF LIGHT CURVE USING LSTM NEURAL NETWORK. JOURNAL OF PHYSICS: CONFERENCE SERIES, 1061:012012, 07 2018.
16. SOVAN BISWAS AND R. VENKATESH BABU. SHORT LOCAL TRAJECTORY BASED MOVING ANOMALY DETECTION. IN PROCEEDINGS OF THE 2014 INDIAN CONFERENCE ON COMPUTER VISION GRAPHICS AND IMAGE PROCESSING, ICVGIP '14, NEW YORK, NY, USA, 2014. ASSOCIATION FOR COMPUTING MACHINERY.
17. ZHENGYING CHEN, YONGHONG TIAN, WEI ZENG, AND TIEJUN HUANG. DETECTING ABNORMAL BEHAVIORS IN SURVEILLANCE VIDEOS BASED ON FUZZY CLUSTERING AND MULTIPLE AUTO-ENCODERS. 2015 IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME),
18. APAPAN PUMSIRIRAT AND LIU YAN. CREDIT CARD FRAUD DETECTION USING DEEP LEARNING BASED ON AUTO-ENCODER AND RESTRICTED BOLTZMANN MACHINE. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, 9, 01 2018.
19. HAOWEN XU, WENXIAO CHEN, N. ZHAO, Z. LI, JIAHAO BU, ZHIHAN LI, Y. LIU, Y. ZHAO, D. PEI, Y. FENG, JIAN JHEN CHEN, ZHAOGANG WANG, AND HONGLIN QIAO. UNSUPERVISED ANOMALY DETECTION VIA VARIATIONAL AUTOENCODER FOR SEASONAL KPIS IN WEB APPLICATIONS. PROCEEDINGS OF THE 2018 WORLD WIDE WEB CONFERENCE, 2018.
20. RAGHAVENDRA CHALAPATHY, A. MENON, AND S. CHAWLA. ANOMALY DETECTION USING ONE-CLASS NEURAL NETWORKS. ARXIV, ABS/1802.06360, 2018.
21. CNN FEATURES WITH BI-DIRECTIONAL LSTM FOR REAL-TIME ANOMALY DETECTION IN SURVEILLANCE NETWORKS [SPRINGER SCIENCE + BUSINESS MEDIA, LLC, PART OF SPRINGER NATURE 2020]
22. REAL-TIME ANOMALY RECOGNITION THROUGH CCTV USING NEURAL NETWORKS [INTERNATIONAL CONFERENCE ON SMART SUSTAINABLE INTELLIGENT COMPUTING AND APPLICATIONS UNDER ICITETM2020].

23. An Efficient Anomaly Recognition Framework Using an Attention Residual LSTM in Surveillance Videos 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution(CC BY) license (https://creativecommons.org/licenses/by/4.0/).
24. Nikolay Laptev, Saeed Amizadeh, and Ian Flint. Generic and scalable framework for automated time-series anomaly detection. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, page 1939–1947, New York, NY, USA, 2015. Association for Computing Machinery.
25. A. Basharat, A. Gritai, and M. Shah. Learning object motion patterns for anomaly detection and improved object detection. In 2008 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, 2008.
26. F. Jiang, Y. Wu, and A. K. Katsaggelos. Detecting contextual anomalies of crowd motion in surveillance video. In 2009 16th IEEE International Conference on Image Processing (ICIP), pages 1117–1120, 2009.
27. Fire and Gun Violence based Anomaly Detection System Using Deep Neural Networks International conference on Electronics and Sustainable Communication Systems - ICESCS 2020
28. Weapon Detection Using YOLO V3 for Smart Surveillance System https://doi.org/10.1155/2021/9975700
29. DeepLearning https://www.researchgate.net/publication/277411157_Deep_Learning?enrichId=rgreqb1c277101047220feb4cef3798840250XXX&enrichSource=Y292ZXJQYWdlOzI3NzQxMTE1NztBUzoyNjc1MzI2MTEzNTQ2MjlAMTQ0MDc5NjE1MzUzNw%3D%3D&el=1_x_2&_esc=publicationCoverPdf
30. What is Machine Learning? | IBM
31. Getting started with Machine Learning - GeeksforGeeks
32. Difference between Deep Learning & Machine Learning (analyticsvidhya.com)
33. Machine Learning vs Deep Learning - DZone AI
34. What are Neural Networks? | IBM
35. What are Convolutional Neural Networks? | IBM

36. Different Types of CNN Architectures Explained: Examples - Data Analytics (vitalflux.com)
37. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Hartwig, A. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv 2017, arXiv:1704.04861.
38. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings
of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
39. Sak H, Senior A, Beaufays F (2014) Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Fifteenth annual conference of the international speech communication association
40. Hochreiter, S.; Schmidhuber, J. Long short-term memory. Neural Comput. 1997, 9, 1735–1780.
41. What are Recurrent Neural Networks? | IBM
42. augmenters.color — imgaug 0.4.0 documentation

الملخص:

العديد من كاميرات المراقبة منتشرة في كل مكان لأغراض أمنية ، لكن الجرائم تحدث في وقت قصير جدًا. المشكلة الكبيرة أن الجريمة لا يتم اكتشافها في نفس الوقت الذي تحدث فيه ، ولكن بعد مراجعة تسجيلات الفيديو لكاميرات المراقبة وقد يستغرق الأمر مرات كثيرة لاكتشاف ما حدث حتى يتمكن حراس الأمن من اكتشاف الجريمة والرد عليها. علاوة على ذلك ، فإن انتباه حراس الأمن يتضاءل مع مرور الوقت مما يفتح المجال أمام المجرمين لبدء مهمتهم بسهولة ونجاح.

نظامنا المقترح هو نظام كامل يمكنه التعرف على السلوك الشاذ على الفور عن طريق إرسال صورة لقطة شاشة لسلوك المغاير والوقت والتاريخ ونوع السلوك ومعرفة المنطقة. من خلال هذه الإخطارات ، يمكن للحارس اتخاذ الإجراء المناسب ، كما سيتم اتخاذ الإجراءات الآلية بناءً على نوع السلوك المغاير.

أيضًا ، يحتوي نظامنا على غرفة تحكم حيث يمكن للمسؤول إضافة رجال الأمن وإزالة رجال الأمن وتعديل تفاصيل رجال الأمن مثل تغيير رجل من منطقة إلى منطقة أخرى ، كما يوفر البث المباشر لمراقبة جميع كاميرات المراقبة في النظام.

لذلك ، سيكون نظامنا المقترح نظام مراقبة ذكيًا للسلوك الإجرامي يلعب دورًا مهمًا في اكتشاف السلوكيات الغير معتادة والتعرف عليها بسرعة وكفاءة وإخطار حراس الأمن واتخاذ الإجراءات التلقائية.