Faculty of Engineering

Helwan University

Eagle Eye

Anomaly Detection System

Graduation project is submitted to Computer Engineering Department in Partial fulfillment of the requirements for the degree of Bachelor of Computer Engineering

BY

Al-Hassan Ali Ahmad Ail
Ashraf Hesham Ahmad Abdslamm
Sarah Gamal El-Deen Mohamed Abdelateef
Salsabeel Ashraf Mohammed Hassan
Yasmine Arafa Kamal Saad

SUPERVISED BY

Prof. Ahmed El-badawi

2022

Faculty of Engineering
Computer Department

Eagle Eye
Anomaly Detection System

By:
Al-Hassan Ali Ahmad Ali
Ashraf Hesham Ahmad
Sarah Gamal Eldeen Abdullatif
Salsabeel Ashraf Mohamed Hassan
Yasmine Arafa Kamal Saad

Supervised by:
Prof. Ahmed ElBadway

1 Table of Contents

Chapter 1
Introduction

1. Introduction

1.1. Overview

Surveillance systems are one of the most important systems in the fields of protection and security of public and private places like stores, companies and even homes. It is very important for everyone to feel safe in every place, therefore police forces and security companies are very interested in surveillance systems. You can see many surveillance cameras in malls and supermarkets you visit daily which is usually used to monitor and record everything happening to protect the place from any suspicious events like stealing, fighting and so on. Also, you can see many surveillance cameras in streets and roads which is usually used to monitor and record the traffic for many purposes like recording vehicles that exceeded the maximum speed, monitoring the status of the roads and checking frequently if there are any accidents and so on.

The market of video surveillance is very big, and it costs billions of USDs worldwide. In many cases of video surveillance systems there is a need for employees to monitor the video and make the suitable action when there is an anomaly event. However, we can't ignore the human error which may prevent to observe the anomaly events. Also, we can't ignore the human reaction speed which may prevent to make the suitable reaction at the suitable time. From business point of view automatic surveillance systems are very efficient because there is no need to hire people to keep an eye on the intended places. However, in many cases, these systems are not efficient and miss many events that should be reported on time. On the other hand, storing the huge amount of data engaged in video surveillance is a difficult problem. One way of solving it is keeping only significant frames or video clips that contain suspicious actions. Here is where Eagle-eye comes. Eagle-eye is an app that aims to automate the surveillance process. The app accepts the video from the camera and detects if there are any suspicious actions. It detects videos containing anomalies and sends notifications to mobile application, it also detects faces in abnormal frames.

1.2. Problem Statement

Nowadays, the safety of individuals is a major concern, the need for developing solutions that enhance safety and security is undeniable. Monitoring and Surveillance systems are being widely deployed in public infrastructure and locations due to its ability to deliver secure and timely information. Almost every public place is now equipped with one, whether you're commuting, shopping, or working, a CCTV camera is always around.

[1] The country that has the most CCTV in the world is China. Given the nation's size, in terms of landmass and population, there is no doubt it topped the list. According to a recent study from Precise Security details, China has a total of 200 million CCTV cameras, and it is still growing. Its major cities and business hubs also lead the rankings

for cities having the highest number of the same surveillance devices. On the other hand, China comes just second to the United States in terms of CCTV count per capita. The world's second-biggest economy has 14.36 cameras installed per 100 people while the United States has 15.28 cameras per hundred.

Here is a list of the top countries with the highest CCTV cameras installed in the world:

1. China – 200 million CCTV cameras
2. United States – 50 million CCTV cameras
3. United Kingdom – 5 million CCTV cameras
4. Japan – close to 5 million CCTV cameras
5. Vietnam – 2.6 million CCTV cameras
6. France – 1.65 million CCTV cameras
7. South Korea – 1.03 million CCTV cameras
8. The Netherlands – 1 million CCTV cameras

[2] The global surveillance technology market is expected to grow from $114.10 billion in 2021 to $130.08 billion in 2022 at a compound annual growth rate (CAGR) of 14%. The market is even expected to reach $214.88 billion in 2026 at a CAGR of 13.4%.

Surveillance Systems provide visual data that deliver various benefits which are deterrence, detection, prevention and response to crimes and disasters, also CCTV camera footage of an incident can be vital in providing 'retrospective' benefits too, as the footage can provide vital evidence that assists police investigations.

The more Surveillance systems installed, the more the need for human operators and manual inspection increases which requires a great number of workforce as well as huge labor costs for observing all day long. They are essentially waiting for something to happen while

they could have been doing something much more important. Accordingly manual inspection requires lots of employees with their constant attention and focus to judge if the captured activities are anomalous or suspicious which is prone to disturbances and tiredness, and thus human errors, since constant observation of these surveillance cameras by humans is a nearly impossible task which surely inefficient and produces errors.

Human error poses a great threat here, especially when suspicious events happen very rapidly and in a short period. What if they miss something? What if something goes unnoticed?

Furthermore, one of the most common problems we face in traditional surveillance systems is the need for huge storage, as these systems are recording 24/7 which consumes a lot of storage and cost. As well as these long-recorded footages are hard to revise and analyses in case of criminal accidents and investigations.

Through this constant observation process there is an important issue, which is privacy, people do not like being observed by someone continuously all the time.

1.3. Goals and objectives

According to what we discussed in the problem statement section, the need for intelligent anomaly surveillance systems is gradually rising although there are some drawbacks.

Our aim is to automate and facilitate the monitoring process of surveillance cameras by implementing an anomaly detection system which can effectively detect fighting events, to reduce the human factor, avoid human errors, save humanity time, and save the workforces wages cost.

With the automation of the monitoring process, we resolved the human factor issue, as we did not need a lot of security monitors, therefore there will be fewer human errors, less wages cost, and more time for humanity.

Moreover, the intelligent surveillance systems became required to detect and memories only frames that contain anomaly activities instead of memories continuous recordings of whole days.

The main objective:

- To reduce the human factor.
- To avoid human error that results from loss of concentration and negligence or unnoticed and fast events.
- Saving human time from observation and revision of long videos when an abnormal event happens.

Chapter 2
Necessary Background

2 Necessary Background

2.1 Overview

In this chapter, we are going to discuss the scientific background of the software and the methods used in understanding the question to get most accurate results.

This chapter discusses the algorithms and techniques we used to build our system. We will illustrate the diverse backgrounds we used in building our project as: Image Processing, Deep Learning, and Machine Learning.

2.2 Deep learning

Deep Learning is a subset of Machine Learning that achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

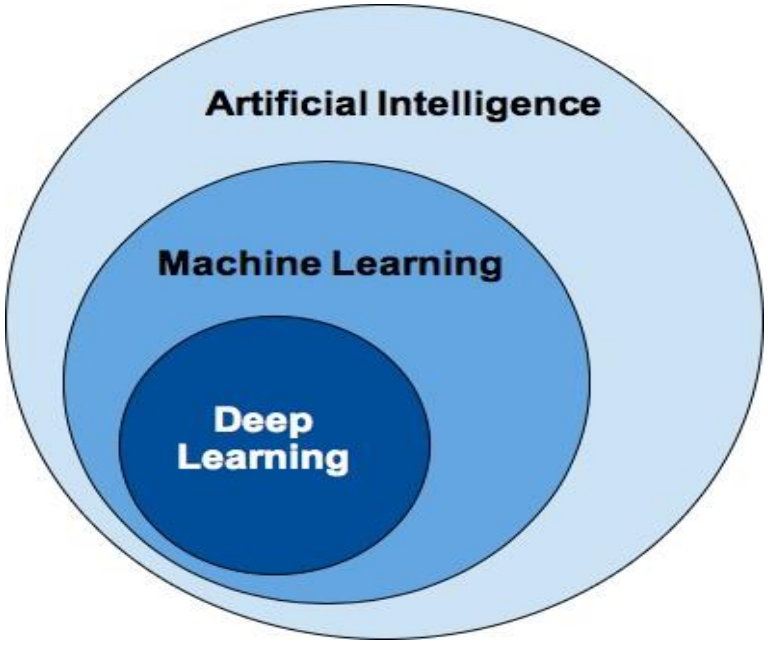


Figure 2.1: Deep Learning, Machine Learning and AI

Elaborately, a deep learning technique learn categories incrementally through its hidden layer architecture, defining low-level categories like letters first then little higher-level categories like words and then higher-level categories like sentences. In the example of image recognition, it means identifying light/dark areas before categorizing lines and then shapes to allow face recognition. Each neuron or node in the network represents one aspect of the whole and together they provide a full representation of the image. Each node or hidden layer is given a weight that represents the strength of its relationship with the output and as the model develops the weights are adjusted.

1. Distinctive Features of Deep Learning

A big advantage with deep learning, and a key part in understanding why it is becoming popular, is that it is powered by massive amounts of data. The "Big Data Era" of technology will provide massive amounts of opportunities for new innovations in deep learning.

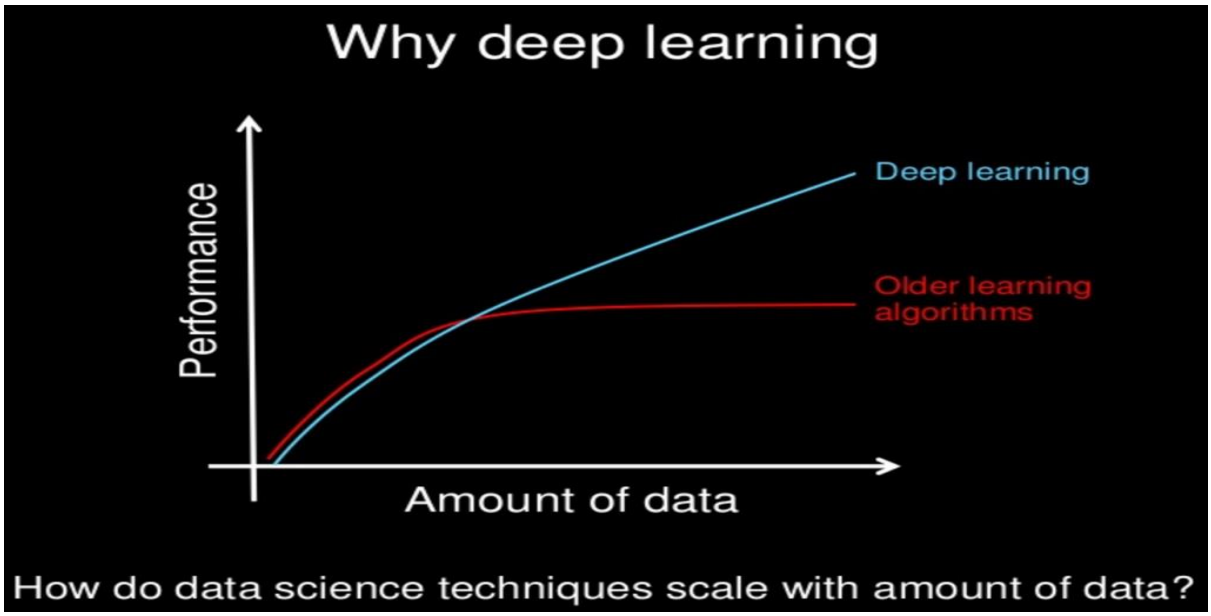


Figure 2.2: Deep learning & Amounts of data

Deep Learning requires high-end machines contrary to traditional Machine Learning algorithms. GPU has become an integral part now to execute any Deep Learning algorithm.

In traditional Machine learning techniques, most of the applied features need to be identified by a domain expert to reduce the complexity of the data and make patterns more visible to learning algorithms to work. The biggest advantage of Deep Learning algorithms as discussed before are that they try to learn high-level features from data in an incremental manner. This eliminates the need of domain expertise and hard-core feature extraction.

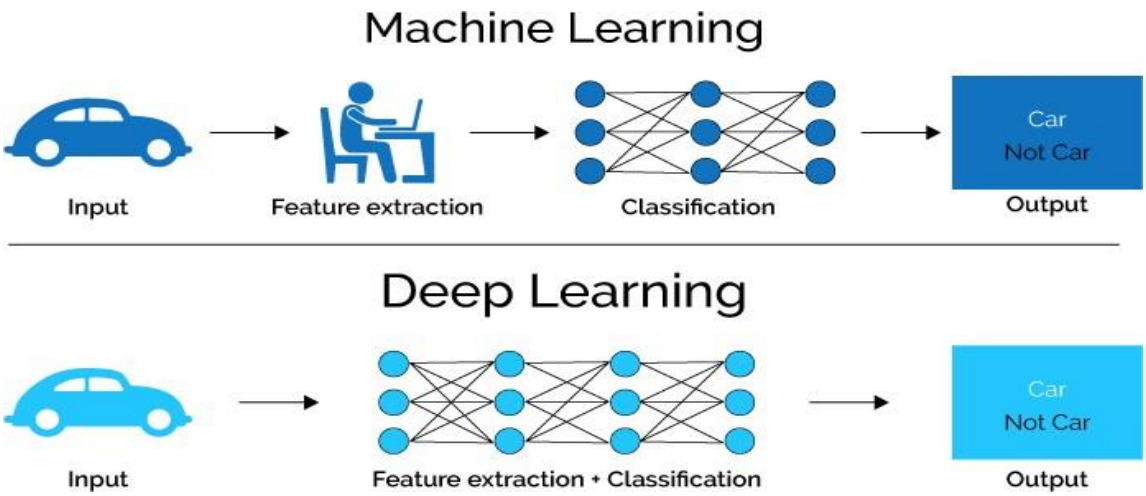


Figure 2.3: Machine learning vs Deep learning

Another major difference between Deep Learning and Machine Learning technique is the problem-solving approach. Deep Learning techniques tend to solve the problem end to end, whereas Machine learning techniques need the problem statements to break down to various parts to be solved first and then their results to be combined at the final stage. For example, for a multiple object detection problem, Deep Learning techniques like **Yolo net** take the image as input and provide the location and name of objects at output. But in usual Machine Learning algorithms like **SVM**, a bounding box object detection algorithm is required first to identify all objects to have the HOG as input to the learning algorithm to recognize relevant objects.

Usually, a Deep Learning algorithm takes a long time to train due to the enormous number of parameters. Popular **ResNet** algorithm takes about two weeks to train completely from scratch. Whereas traditional Machine Learning algorithms take few seconds to a few hours to train. The scenario is completely reverse in testing phase. At test time, Deep Learning algorithm takes much less time to run. Whereas, if you compare it with k-nearest neighbors (a type of machine learning algorithm), test time increases on increasing the size of data. Although this is not applicable on all machine learning algorithms, as some of them have small testing times too.

2. When to use Machine Learning?

   - When you cannot code the rules: Many human tasks (such as recognizing whether an email is spam or not spam) cannot be solved using a simple (deterministic), rule-based solution. A large number of factors could influence the answer. When rules depend on too many factors and many of these rules overlap or need to be tuned very finely, it soon becomes difficult for a human to accurately code the rules. You can use ML to effectively solve this problem.

   - When you cannot scale: You might be able to manually recognize a few hundred emails and decide whether they are spam or not. However, this task becomes tedious for millions of emails. ML solutions are effective at handling large-scale problems.

2.3 Supervised learning

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from **labeled training** data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the **supervisory signal**).

A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for **unseen instances**. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Regression and **classification** are categorized under the same umbrella of supervised machine learning. Both share the same concept of utilizing known datasets (referred to as training datasets) to make predictions.

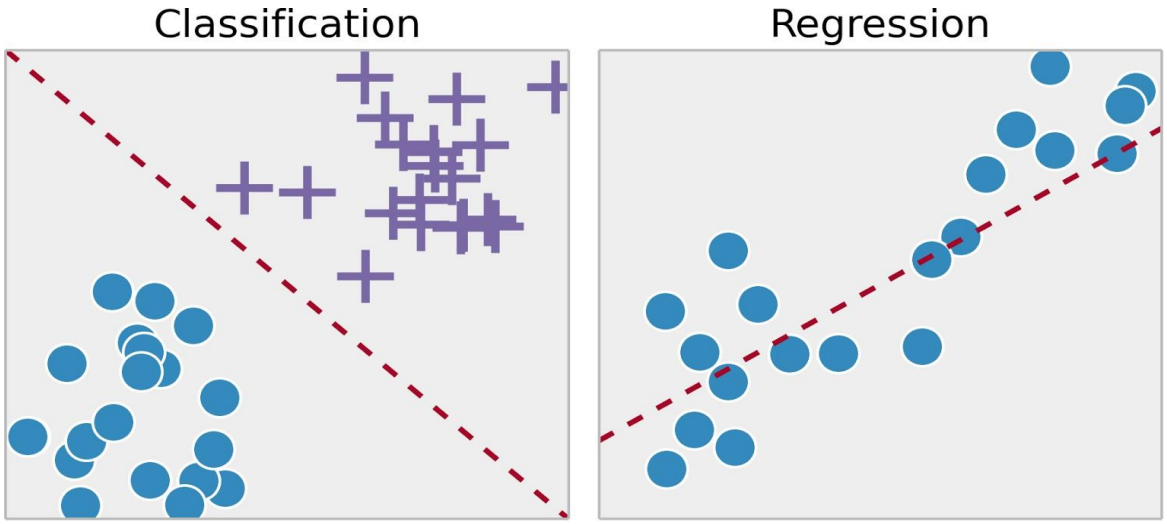


Figure 2.4: Classification vs Regression

In supervised learning, an algorithm is employed to learn the mapping function from the input variable (x) to the output variable (y); that is **y = f(X)**.

The objective of such a problem is to approximate the mapping function (f) as accurately as possible such that whenever there is a new input data (x), the output variable (y) for the dataset can be predicted. (**Figure 2.3**) shows that the main difference between them is that the output variable in regression is numerical (or continuous) while that for classification is categorical (or discrete).

Examples of the common regression algorithms include linear regression, Support Vector Regression **(SVR)**, and regression trees.

Examples of the common classification algorithms include logistic regression, Naïve Bayes, decision trees, and K-Nearest Neighbors **(KNN)**.

2.4 Unsupervised learning

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.

1. Common unsupervised learning approaches

Unsupervised learning models are utilized for three main tasks clustering, association, and dimensionality reduction. Below we will define each learning method and highlight common algorithms and approaches to conduct them effectively.

1. Clustering

Clustering is a data mining technique which groups unlabeled data based on their similarities or differences. Clustering algorithms are used to process raw, unclassified data objects into groups represented by structures or patterns in the information. Clustering algorithms can be categorized into a few types, specifically exclusive, overlapping, hierarchical, and probabilistic.

2. Exclusive and Overlapping Clustering

Exclusive clustering is a form of grouping that stipulates a data point can exist only in one cluster. This can also be referred to as "hard" clustering. The K-means clustering algorithm is an example of exclusive clustering.

1. K-means clustering is a common example of an exclusive clustering method where data points are assigned into K groups, where K represents the number of clusters based on the distance from each group's centroid. The data points closest to a given centroid will be clustered under the same category. A larger K value will be indicative of smaller groupings with more granularity whereas a smaller K value will have larger groupings and less granularity. K-means clustering is commonly used in market segmentation, document clustering, image segmentation, and image compression.

Overlapping clusters differs from exclusive clustering in that it allows data points to belong to multiple clusters with separate degrees of membership. "Soft" or fuzzy k-means clustering is an example of overlapping clustering.

- **Autoencoders**

  Autoencoders leverage neural networks to compress data and then recreate a new representation of the original data's input. Looking at the image below, you can see that the hidden layer specifically acts as a bottleneck to compress

the input layer prior to reconstructing within the output layer. The stage from the input layer to the hidden layer is referred to as “encoding” while the stage from the hidden layer to the output layer is known as “decoding.”

Variants exist, aiming to force the learned representations to assume useful properties. Examples are regularized autoencoders (Sparse, Denoising and Contractive), which are effective in learning representations for subsequent classification tasks, and Variational autoencoders, with applications as generative models. Autoencoders are applied to many problems, including facial recognition, feature detection, anomaly detection and acquiring the meaning of words. Autoencoders are also generative models which can randomly generate new data that is like the input data (training data).

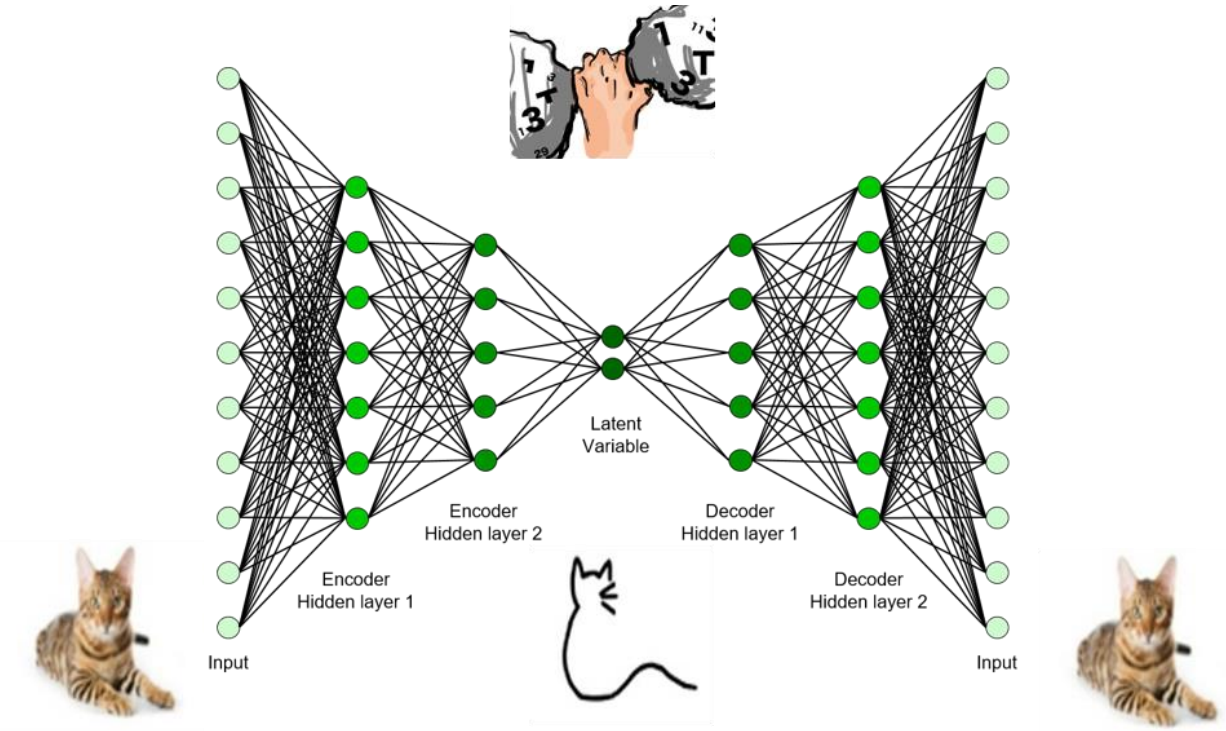


Figure 2.5: Autoencoder

2. Unsupervised vs. supervised

Unsupervised learning and supervised learning are frequently discussed together. Unlike unsupervised learning algorithms, supervised learning algorithms use labeled data. From that data, it either predicts future outcomes or assigns data to specific categories based on the regression or classification problem that it is trying to solve. While supervised learning algorithms tend to be more accurate than unsupervised learning models, they require upfront human intervention to label the data appropriately. However, these labelled datasets allow supervised learning algorithms to avoid computational complexity as they do not need a large training set to produce intended outcomes. Common regression and classification techniques are linear and logistic regression, naïve bayes, KNN algorithm, and random forest.

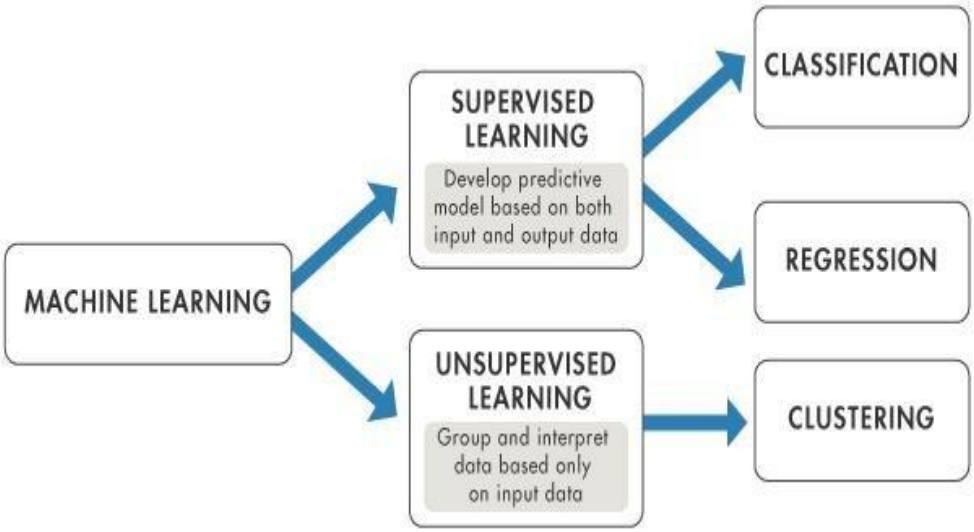


Figure 2.6: Unsupervised vs. supervised

2.5 Loss Function

Loss function is one of the most important concepts in deep learning. A loss function is incredibly simple: it is a method of evaluating how well your algorithm models your dataset. If your predictions are off, your loss function will output a higher number. If they are good, it will output a lower number. As you change pieces of your algorithm to try and improve your model, your loss function will tell you if you are getting anywhere.

Loss function or cost function is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event. An optimization problem seeks to minimize a loss function. An objective function is either a loss function or its negative (reward function or utility function), in which case it is to be maximized. It represents how far off your desired start you are.

Some of the most common loss functions used are:

1. Regression Losses
   3. Mean squared Error (MSE):

   MSE is the most common loss function in machine learning. It is also the most intuitive loss function.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

   - Mean Absolute Error (MAE) / L1 Loss

     MAE is measured as the average of sum of absolute differences between predictions and actual observations.

$$MAE = \frac{\sum_{i=1}^{n} \mid y_i - \hat{y}_i \mid}{n}$$

- Mean Bias Error (MBE)

  This is much less common in machine learning domain as compared to its counterpart. This is same as MSE with the only difference that we do not take absolute values.

$$MBE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}$$

2. Classification Losses

   - Hinge Loss/Multi class SVM Loss

     In simple terms, the score of correct categories should be greater than the sum of scores of all incorrect categories by some safety margin (usually one). And hence hinge loss is used for maximum-margin classification, most notably for support vector machines. Although not differentiable, it is a convex function which makes it easy to work with usual convex optimizers used in machine learning domain.

$$SVMLoss = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$$

   - Cross Entropy Loss

     This is the most common setting for classification problems. Cross-entropy loss increases as the predicted probability diverges from the actual label.

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

2.6 Optimization Algorithms

Optimization techniques are used to minimize the loss of a model during its training process by finding the updates necessary to add to the model values that would result in the decrease of the training loss.

Some of the most popular optimization algorithms are:

1. **Gradient descent**

   It is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. If, instead, one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as gradient ascent.

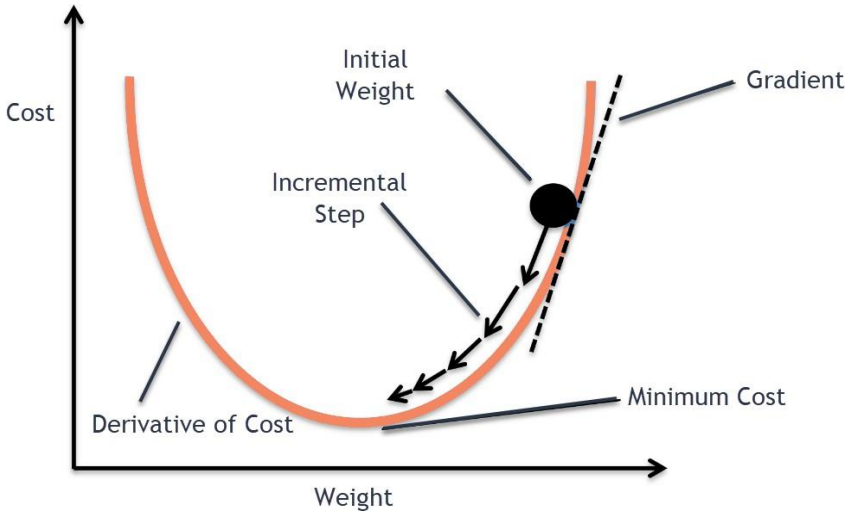


Figure 2.7: Gradient Descent

The size of these steps is called the learning rate. With a high learning rate, we can cover more ground each step, but we risk overshooting the lowest point since the slope of the hill is constantly changing. With an extremely low learning rate, we can confidently move in the direction of the negative gradient since we are recalculating it so frequently. A low learning rate is more precise, but calculating the gradient is time-consuming, so it will take us an exceptionally long time to get to the bottom.

2. Stochastic Gradient Descent (SGD)

Gradient descent is a famous algorithm for gradient-based optimization. It is used to find optimal parameters for minimizing or maximizing functions by updating the parameter in the direction (gradient) of the required optimization objective (minimization or maximization). SGD is an incremental variant of gradient descent where updating the parameter takes place incrementally using batches of the training set rather than using the whole training set for just one parameter update.

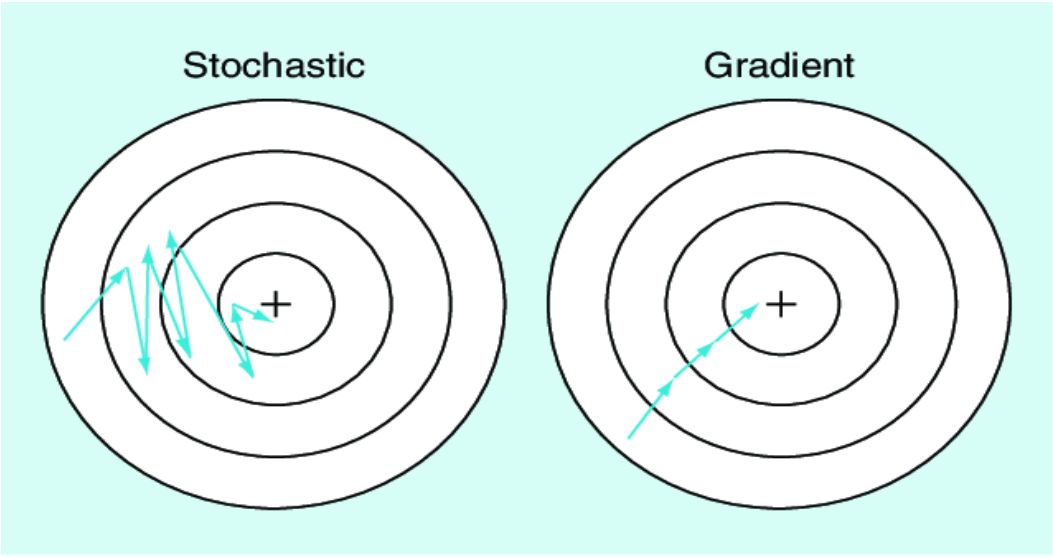


Figure 2.8: Gradient Descent vs Stochastic Gradient Descent

3. Momentum

Further proposals include the momentum method, which appeared in Rumelhart, Hinton, and Williams' seminal paper on backpropagation learning. Stochastic gradient descent with momentum remembers the update Δ w at each iteration and determines the next update as a linear combination of the gradient and the previous update.

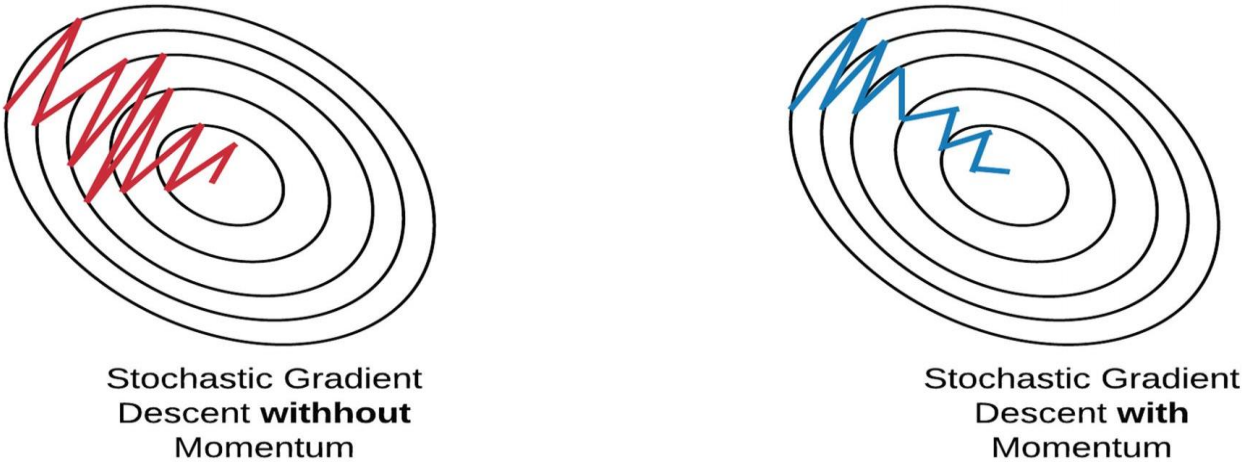


Figure 2.8: Stochastic Gradient Descent without Momentum vs Stochastic Gradient Descent with Momentum

4. Averaging

Averaged stochastic gradient descent, invented independently by Ruppert and Polyak in the late 1980s, is ordinary stochastic gradient descent that records an average of its parameter vector over time. When optimization is done, this averaged parameter vector takes the place of w.

5. Adaptive Gradient (AdaGrad)

It is a modified stochastic gradient descent with per-parameter learning rate, first published in 2011. Informally, this increases the learning rate for more sparse parameters and decreases the learning rate for less sparse ones. This strategy often improves convergence performance over standard stochastic gradient descent in settings where data is sparse and sparse parameters are more informative. Examples of such applications include natural language processing and image recognition.

6. AdaDelta

It is an extension of AdaGrad that aim to mitigate its aggressive. It gradually decreases learning rate. Instead of accumulating all past squared gradients, AdaDelta restricts the window of accumulated past gradients to some fixed size window. Instead of inefficiently storing all previous squared gradients, the sum of gradients is recursively defined as a decaying average of all past squared gradients.

7. Adaptive Moment Estimation (Adam)

It is yet another method that computes adaptive learning rates for each parameter. Like AdaDelta and RMSprop it stores the previous squared gradients, but in addition to this it stores an exponentially decaying average of past gradients like the momentum.

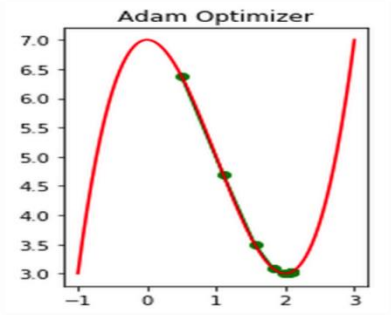


Figure 2.9: Adam Optimizer

8. Nesterov's Accelerated Gradient (Nesterov)

The intuition behind Nesterov is to have a smart way to approach local optima. For example, in minimization Instead of blindly pushing parameters down the hill with acceleration, we need to be smart and not to accelerate them to the point they get past the local minimum. I.e., the parameters need to slow down before reaching the local optimum point. Nesterov accelerated gradient is a way to give the momentum term this kind of foresight.

9. RMSProp

RMSProp (for Root Mean Square Propagation) is also a method in which the learning rate is adapted for each of the parameters. The idea is to divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. RMSProp has shown excellent adaptation of learning rate in different applications. RMSProp can be seen as a generalization of RMSprop and is capable to work with mini-batches as well as opposed to only full-batches.

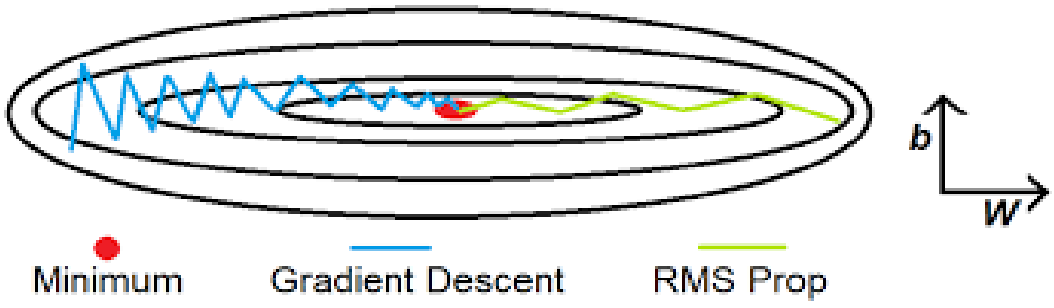


Figure 2.10: Minimum vs. Gradient Descent vs. RMS Prop

2.7 Neural Networks

A neural network is an information processing paradigm that is loosely inspired biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of many highly interconnected processing elements (neurons) working in unison to solve specific problems.

Neural networks, like people, learn by example thus they are supervised learning. Neural networks are configured for a specific application, such as pattern recognition or data classification, through a learning process.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new unseen situations of interest and answer "what if" questions.

As illustrated in figure 3.9, a typical neural network consists of an input layer, hidden layer(s), and an output layer. Each layer consists of one or more nodes. Input layer is passive meaning it does not change the input. In comparison the hidden and output layer are active meaning they modify their input. The input flows through the network until it reaches the output layer.

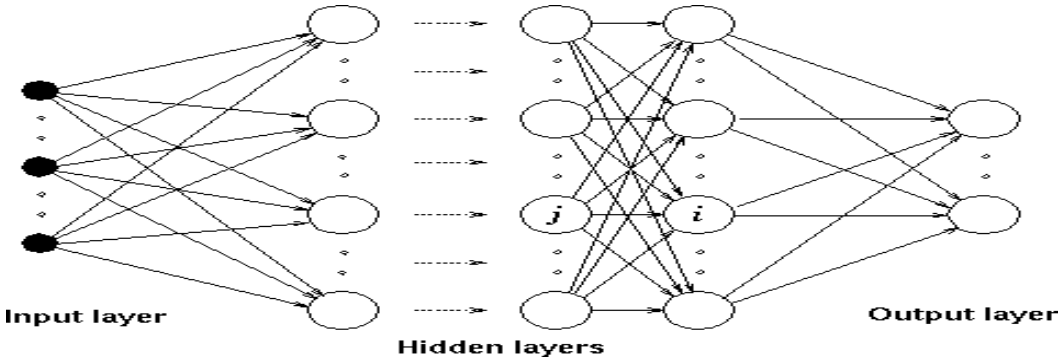


Figure 2.11: neural network architecture

2.8 Feedforward network

A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle. As such, it is different from recurrent neural networks.

The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

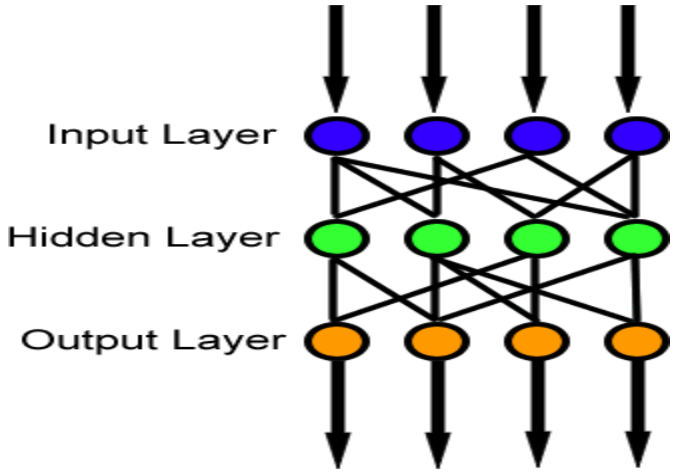


Figure 2.12: Feedforward network

Chapter 3
Previous work

3 Previous work

3.1 Previous work

Anomaly detection is considered one of the most challenging and long-standing dilemmas in computer vision, as real-world anomalous events are complicated and diverse, the boundary between what is considered normal and anomalous behaviors is often ambiguous and hard to be classified, listing all of the possible anomalous events is such a difficult task. Additionally, the same behavior could be considered a normal or an anomalous behavior under different conditions, therefore there are different approaches that have been proposed to detect anomalies to enhance the detection ability.

Anomaly detection algorithms established on different machine learning models and abnormal situations have shown great progress in recent years, as [3],[4], [5],[6] and [7].Various deep learning models were adopted by different studies such as LSTM (Long-Short Term Memory)[8],[9]and [10], also autoencoder is a famous implementation that was adopted by many such as [11] ,[12] and [13] and another technique OC-NN in [14] and Markov model [15] .For video surveillance applications, there are several attempts to detect violence or aggression such as [16],[17],[18]and[19].

There are also different techniques to preprocess images and feature extraction were implemented such as Gao et a [16]. introduced in crowd videos violent flow descriptors to detect violence, while Kooij et al. [17] didn't just use video but also audio data to detect aggressive actions in surveillance videos also Datta et al [18] the detection of human violence was done by exploiting motion and limbs orientation of people and Mohammadi et al. [19] proposed a new behavior heuristic-based approach to differentiate between violent and non-violent videos.

Beyond violent and non-violent patterns distinction, [4, 5] presented the use of tracking to model the normal motion of people and detect deviation from that normal motion as an anomaly, however it’s quite difficult to obtain reliable tracks, several approaches avoid tracking and learn global motion patterns through histogram-based methods such as [20] or motion patterns [21].

The success of sparse representation and dictionary learning methods in several computer vision problems encouraged [22], [23] to use sparse representation to learn the dictionary of normal behaviors. In the testing phase, the patterns with large reconstruction errors are treated as anomalous events. Deep learning algorithms have shown great success to classify images as a result several approaches have been proposed for video action classification [24],[25]. However, obtaining annotations for training is burdensome and laborious especially for videos. Autoencoders are a kind of deep learning models [18] used to learn the model of normal behaviors and reconstruction loss obtained to detect anomalies.

Several techniques were adopted to perform the preprocessing of images, Optical flow is a famous feature used in the pre-processing that was used in [26], [27], [28], [29], [30], [31],

[32], [33]. Time-series has also been looked at in the anomaly detection topic such as [34], however, there is minimal work done on applying time-series on surveillance videos.

The majority of the work done so far can be divided to two categories: -Target-based- non-Target-based.

Target-Based can be thought of as Appling some preprocessing algorithms to extract the objects in the frame such as pedestrians and cars authors like [35], [31], adopted this approach, while authors in [36], [26], [28] do not focus on extracting objects, rather focus on training with the processed fragments, images, or videos. Our method is non-target-based, where the only focus is on the information of motion among the frames.

3.2 Proposed solution

Although the above approaches can achieve satisfactory results, they still encounter some challenges like balancing the time to process one frame (frame rate), the required number of frames to make one prediction, and the training time. The contributions we present can be outlined briefly as:

1) We combined the concept of time series with the video anomaly detection, by obtaining Combined-Difference-Image (CDI). The CDI eliminated the background to ensure the frame energy's accuracy for describing the motion of moving objects.

2) We applied Combined-Difference-Image (CDI) to extract features and information entropy to get the frame energy, then combined all the frame energies of the video as the Video-Energy-Vector (VEV). The information entropy successfully converted the 3D spatial-temporal feature map into the feature vector, significantly reducing the computing time.

3) The VEVs are trained by a two-class SVM to predict whether the video is abnormal or not. The training efficiency of SVM is ensured due to the low dimensionality of VEV

4) We also implemented detection of faces present in the anomalous frame.

3.3 System block diagram

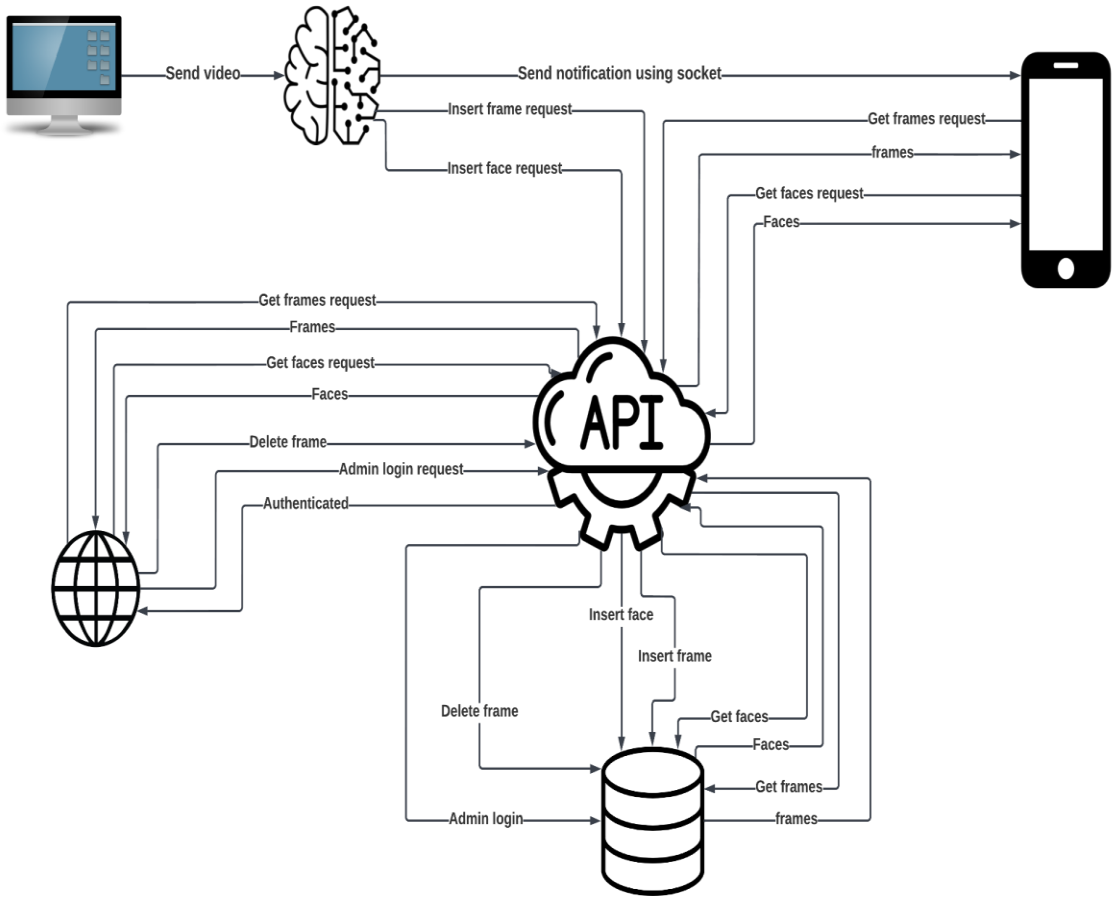


Figure 3.1: System block diagram

Chapter 4
Design specifications

4 Design specifications

4.1 Analysis and Requirements

4.1.1 Functional Requirements

ID	Label	Requirements
1	Record live video	The desktop application allows the security monitor to record live video and analyze it from anomaly events.
2	Upload recorded video	The desktop application allows the security monitor to upload a video and analyze it from anomaly events.
3	Anomaly detection	The desktop application can pass a video to the ML model to process it, and in case an anomaly event is detected the server sends notification to the mobile application with the first anomaly frame, then stores it locally and in the database.
3	Get notified	The security guard should be logged in as a user to receive notification in case of an anomaly event.
4	Displaying Anomaly frames history	The user should be logged in as a normal user or admin to display Anomaly frames history.
5	Displaying faces history	The user should be logged in as a normal user or admin to display faces history.
6	delete anomaly	The user should be logged in as admin to have the privilege of deleting.

7	Storage saving	The task of anomaly frames deletion should be scheduled in the system using Windows Task Scheduler (locally) OR cPanel (hosting) to execute the task daily which checks frames older than 30 days then delete them.

4.1.2 Non-functional Requirements

ID	Label	Requirements
8	Usability	The system interface should be easy to use and remember how to use, easy to modify and maintain.
9	Scalability	The system Should fit in heavier traffic intersections.
10	Reliability	System should perform the task for which it was designed or intended correctly, for a specified time and in a specified environment.
11	Availability	System should be available to perform its task and function under normal conditions
12	Maintainability	System should be easily Maintainable, to handle exceptions and new requirements, also to be flexible with changes through periodic maintenance, as it can handle the bugs or failure occurring in the system while running.
13	Supportability	System should support users to access all the interfaces easily and be capable to update and maintain in future.
14	Security	System should be safe and ensure security concepts. It will guarantee secure transfer of data as it provides different levels of privileges by ensuring authentication.

4.1.3 Functional Requirements Specification

4.1.3.1 Stakeholders

1. Organization and authorities
2. Security guards
3. Security monitor
4. Security manager
5. Software maintenance engineer

4.1.3.2 Actor and Goals

4. Actor
    1. Camera (initiating)
    2. Security guard (participating)
    3. Security manager (initiating)
5. Goals
    1. Capture video of running events.
    2. Detecting and saving anomaly footage.
    3. Reducing crime rate.

4.2 System Design

4.2.1 Use Case

- Use Case Description

Use case	Description	Actors	Conditions
Authentication	Our system has multiple authentication feature to support different levels of privileges. Such as the security manager must login as admin to have plus privileges than the normal users (guards).	Security guards, Security manager	____
Record	Security monitor can record video	Security monitor	Record
stop	Security monitor can stop video	Security monitor	stop
check	Security monitor can upload video to check if its abnormal or not	Security monitor	check
Add guards	The Security manager can add new guards (users) to the system.	Security manager	The actor must be logged in as an admin.
Display frames history	The Security manager and guards can display frames history.	Security guards, Security manager	Actors must be logged in.
Display faces history	The Security manager and guards can display faces history.	Security guards, Security manager	Actors must be logged in.

Delete any frame	The Security manager can delete frames.	Security manager	The actor must be logged in as an admin.
Automatic delete frames history	Executing scheduled task of deleting frames history older than 30 days to save storage.	Server	Frames happened at field must be older than 30 days
Receive notifications	Security guards can receive notifications.	Security guards	The actor must be logged in as a user.
Find face	Select the specific face	Security guard and Security manager	Actors must be logged in.
Find frame	Select the specific frame	Security guard and Security manager	Actors must be logged in.

- Use Case Diagram

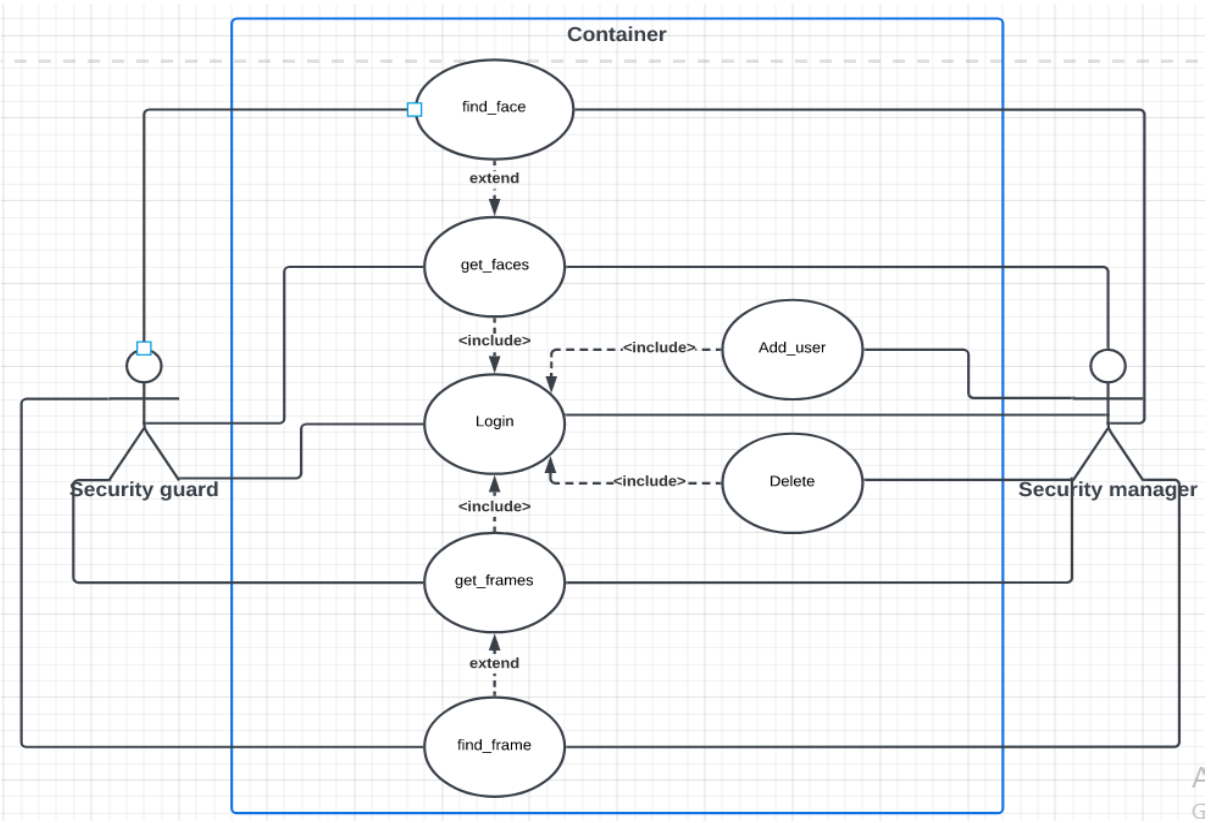


Figure 4.1: Security guard and security manager

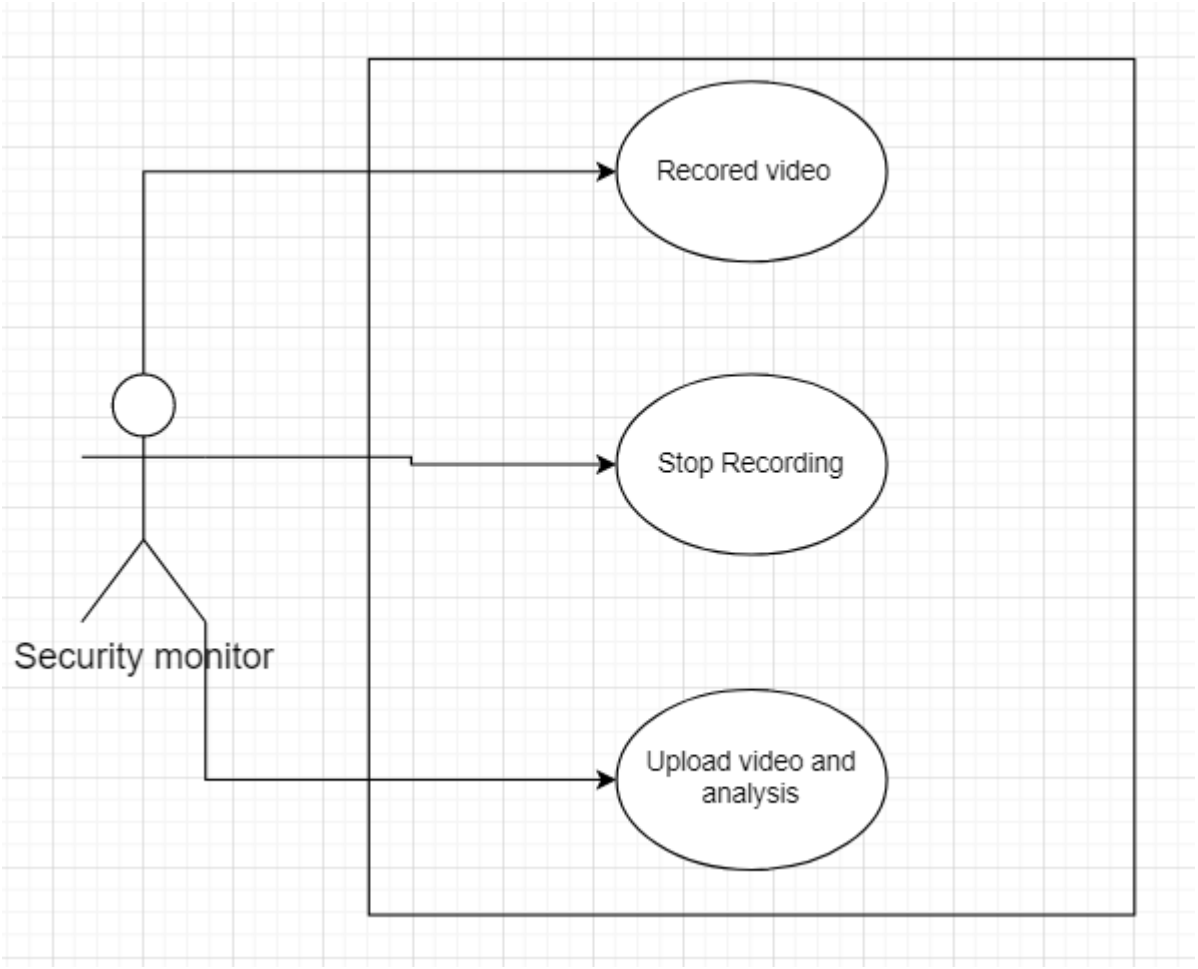
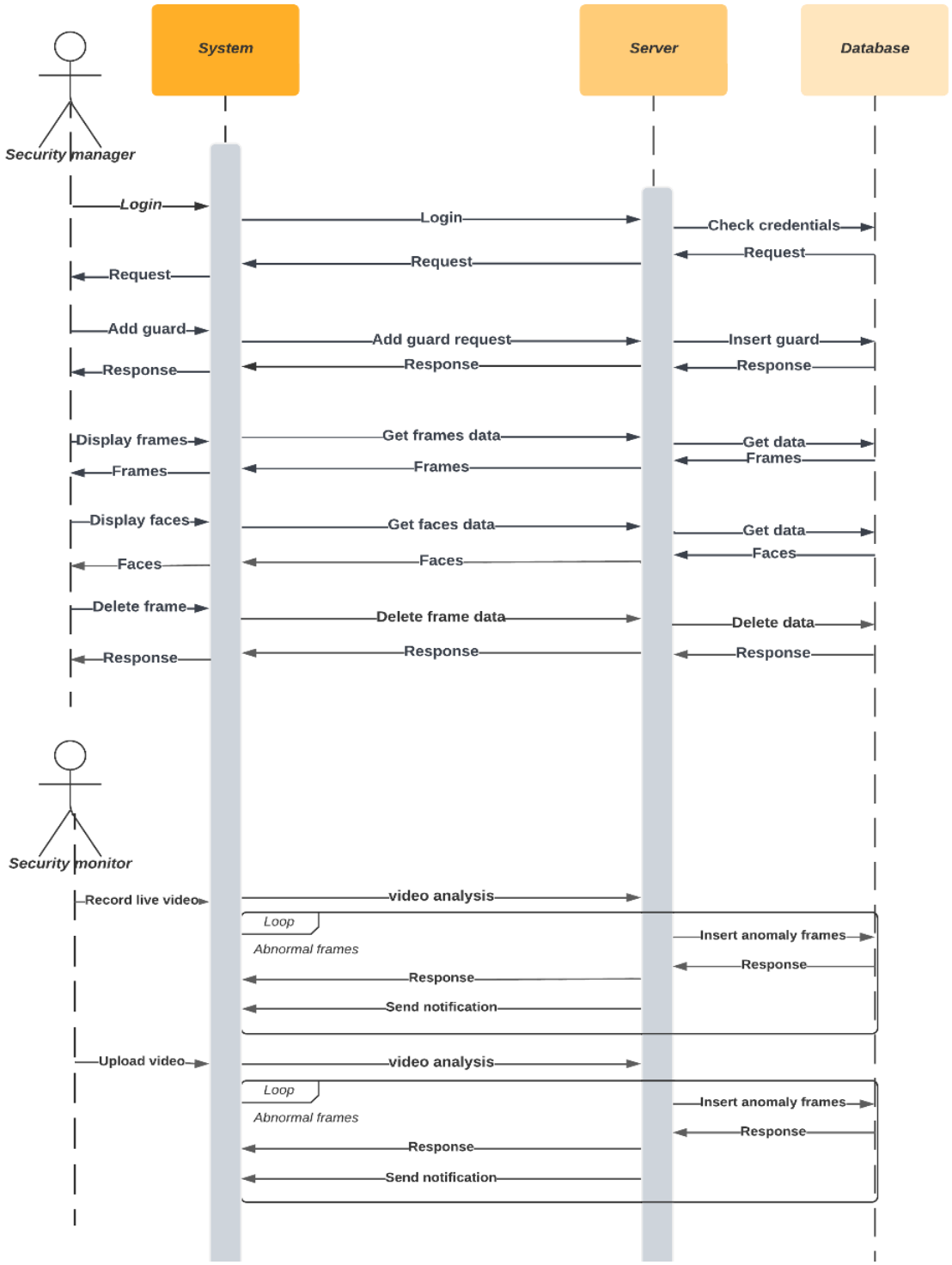


Figure 4.2: security monitor

4.2.2 System Sequence Diagram



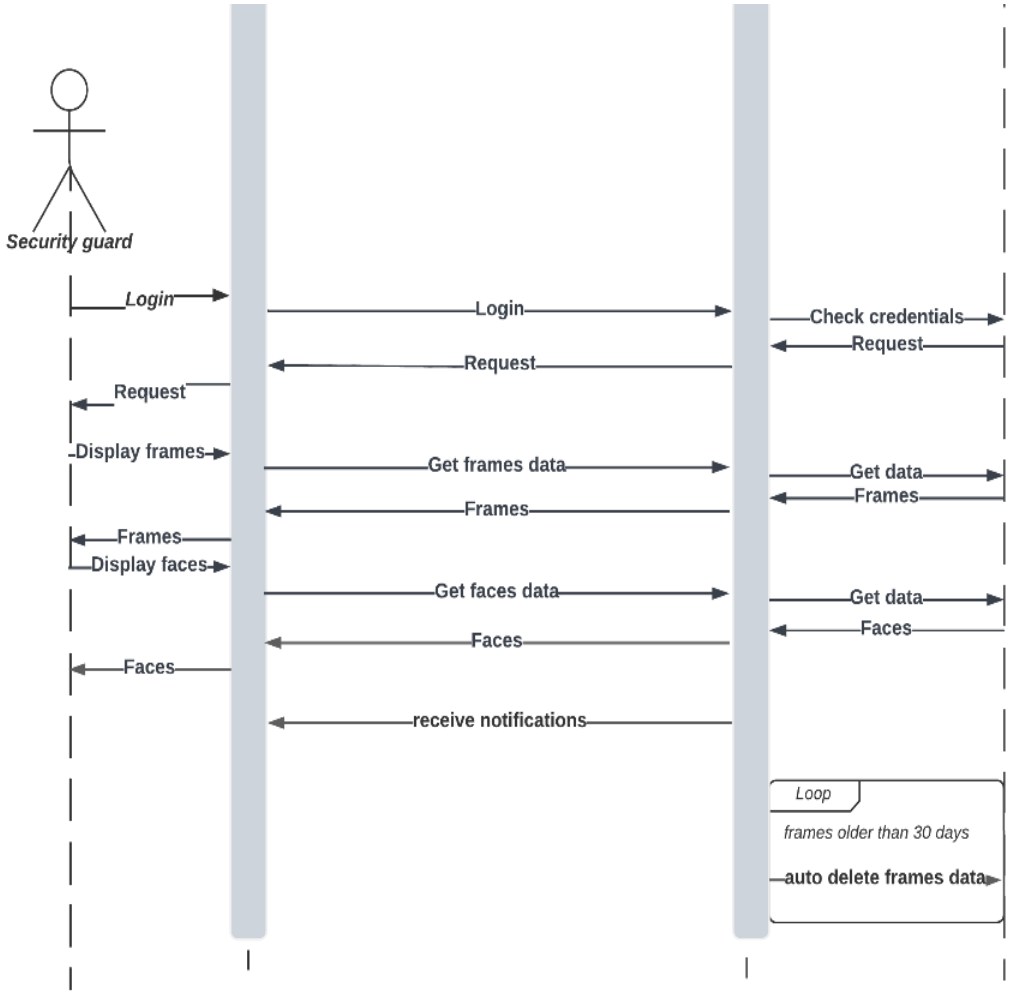


Figure 4.3: System Sequence Diagram

Chapter 5
Intelligence

5 Intelligence

5.1 Methodology

The workflow is shown as Fig. 5.1. The first stage is extracting the feature map of frames, calculating the CDI, obtaining the frame energy and VEV; the second stage is training VEVs on the two-class SVM to predict whether the video has abnormal events.

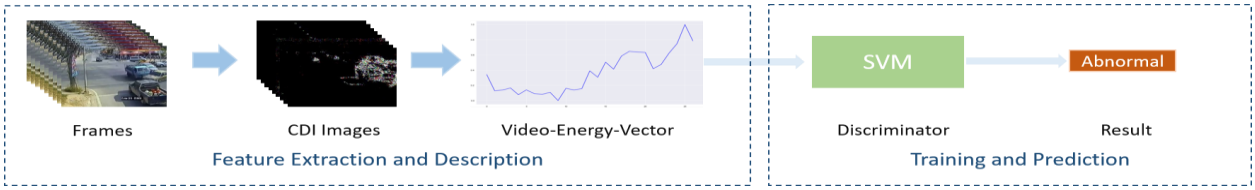


Fig. 5.1. Workflow of our real-time anomaly detection algorithm

5.1.1 Feature Extraction Based on CDI

Since our main task is analyzing the feature effectiveness on detecting five anomaly types, we transformed the original frame F_n to feature maps M_n, n is the number of frames in one video and is equal to 30 in our experiment. We extracted 13 kinds of different feature maps as the light gray blocks shown in Fig. 5.2. Gray transformed the original images into gray images as their feature maps, which is the baseline of these features. The saliency map is based on [19], and we extracted it from both RGB and HSV space (SA-RGB and SA- HSV). We recorded the magnitude and angle of optical flow images as OF-Mag and OF-Ang. We separately calculated the AUC scores of their predictions and compared them in the Feature-Event-Table, which will be shown in Section V.

According to [12], we got the Backward-Difference-Image (BDI) and Forward-Difference-Image (FDI) by the following function:

$$BDI_k = |M_{k-1} - M_k| \quad (1)$$

$$FDI_k = |M_k - M_{k+1}| \quad (2)$$

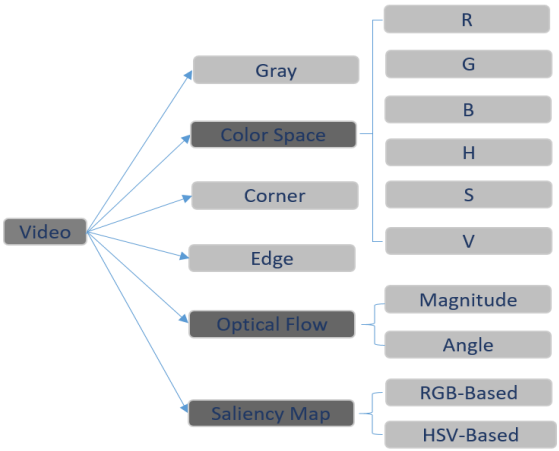


Fig. 5.2: Features Maps

The range of k is equal to (1, n-2) because the first frame does not have BDI, and the last frame does not have FDI. We further optimized them by setting the pixel values which are less than the *threshold* as 0. Here i = 0...w-1, j = 0...h-1, w and h are the width and height of the BDI and FDI:

$$BDI_k^{i,j} = \begin{cases} BDI_k^{i,j} & BDI_k^{i,j} > threshold \\ 0 & else \end{cases} \quad (3)$$

$$FDI_k^{i,j} = \begin{cases} FDI_k^{i,j} & FDI_k^{i,j} > threshold \\ 0 & else \end{cases} \quad (4)$$

Then the CDI is calculated by:

$$CDI_k = BDI_k \bigcup FDI_k \quad (5)$$

According to the process of this frame feature extractor, we can assume that when the moving speed of objects suddenly increases or decreases, which is usually related to the potential abnormal events, the number and value of non-zero pixels in CDI will change correspondingly, which could interpret the unusual changes in the video. As shown in Fig. 5.3, in the target frame, the driver falls out of the car window due to the car accident, and CDI has successfully removed the background, located the abnormal objects, and extracted the motion.

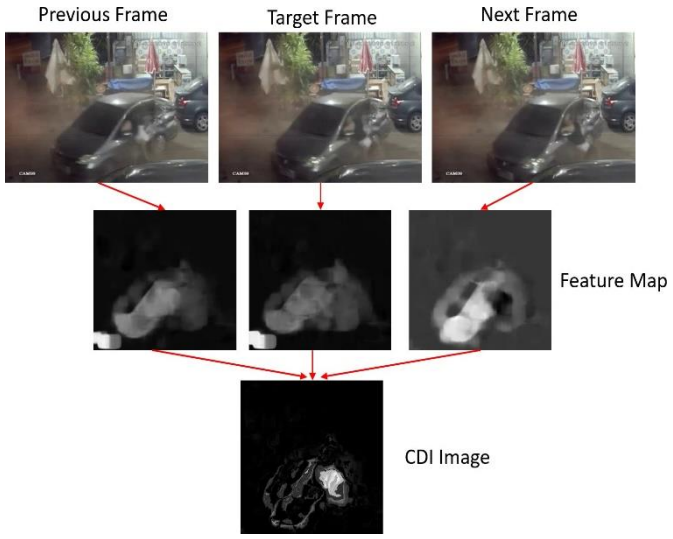


Fig. 5.3: Visualization of Feature Extractor

5.1.2 Feature Description Based on Information Entropy

Dimensionality reduction is a common operation in video training because the high feature dimension contains redundant information and greatly increases the training time. Here we converted a feature map into one single value, named Frame Energy (FE), by extracting its information entropy. Therefore, the dimension of the video descriptor is only depended on the number of frames, and we named it as Video-Energy-Vector (VEV).

The calculation of FE_k of CDI image CDI_k is shown as follow, first we created a histogram H_k to count the number of different values in each CDI: $H_k = \{ \eta_{0k}, \eta_{1k}, \eta_{2k}, \ldots \eta_{\beta k}\}$, here β= 255, as we scaled the pixels in CDI to integers from 0 to 255. Then H_k is replaced by probability vector as: $P = \{p_{0k}, p_{1k}, p_{2k}, \ldots p_{pk}\}$:

$$\rho_{ik} = \frac{\eta_{ik}}{\sum_{m=0}^{\beta} \eta_{mk}} \qquad (6)$$

Then the F Ek is calculated as:

$$FE_k = -\sum_{m=0}^{\beta} \rho_{mk} log(\rho_{mk}) \qquad (7)$$

VEV is the set of FEk as: $V = \{FE1, FE2, \ldots Fen\text{-}2\}$.

5.1.3 SVM Training and Voting System

We trained the five datasets on two-class SVM, respectively. We recorded the AUC scores based on 5-fold cross-validation as our estimation criteria of Feature-Event-Table. We also proposed a voting system to reduce the false alarm rate:

The top 3 features based on AUC are described as: $Li = \{li1, li2, \ldots lit\}$, i = 0, 1, 2, t is the number of data in the test set. We gave the top 1, top 2 and top 3 features the weights: 40%, 30% and 30%, and set the threshold as 0.6. In other words, only when both the top 1 and another feature vote for positive together, will the output be positive. Then the final prediction $L = \{l_1, l_2, \ldots l_t\}$ is voted as:

$$L = 0.4L_0 + 0.3L_1 + 0.3L_2 \qquad (8)$$

$$l_i = \begin{cases} 1 & l_i > 0.6 \\ 0 & else \end{cases} \qquad (9)$$

5.1.4 Face Detection using Haar Cascades

- Basics

  Object Detection using Haar feature-based cascade classifiers is an effective method proposed by Paul Viola and Michael Jones in the 2001 paper, "Rapid Object Detection using a Boosted Cascade of Simple Features". It is a machine learning based approach in which a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

  Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting the sum of pixels

under the white rectangle from the sum of pixels under the black rectangle as shown in fig15.

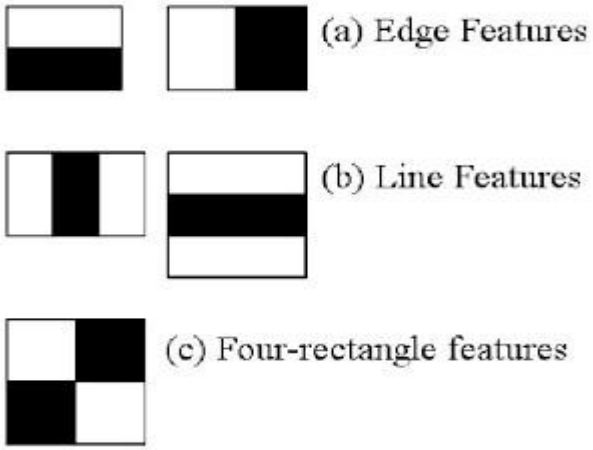


Figure 5.4: haar_features

Now all possible sizes and locations of each kernel are used to calculate plenty of features. For each feature calculation, we need to find the sum of the pixels under the white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of the sum of the pixels, how large may be the number of pixels, to an operation involving just four pixels.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks, or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved by Adaboost.

Figure 5.5

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. (The process is not as simple as this Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then again same process is done. New error rates are calculated. Also new weights. The process is continued until required accuracy or error rate is achieved or required number of features are found).

Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

So now you take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. Wow... Wow... Isn't it a little inefficient and time consuming? Yes, it is. Authors have a good solution for that.

In an image, most of the image region is non-face region. So, it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Don't process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.

For this they introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very a smaller number of features). If a window fails the first stage, discard it. We don't consider remaining features on it. If it passes, apply the second stage of features, and continue the process. The window which passes all stages is a face region. How is the plan!!!

Authors' detector had 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in first five stages. (Two features in the above image is actually obtained as the best two features from Adaboost). According to authors, on an average, 10 features out of 6000+ are evaluated per sub-window.

So, this is a simple intuitive explanation of how Viola-Jones face detection works. Read paper for more details.

- Haar-cascade Detection in OpenCV

  Here we will deal with detection. OpenCV already contains many pre-trained

classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder. Let's create a face and eye detector with OpenCV.

We use the function: detectMultiScale (image, objects, scaleFactor = 1.1, minNeighbors = 3, flags = 0, minSize = new cv.Size(0, 0), maxSize = new cv.Size(0, 0))

Parameters

image matrix of the type CV_8U containing an image where objects are detected.

Objects vector of rectangles where each rectangle contains the detected object. The rectangles may be partially outside the original image.

scaleFactor parameter specifying how much the image size is reduced at each image scale.

minNeighbors parameter specifying how many neighbors each candidate rectangle should have to retain it.

Flags parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade.

minSize minimum possible object size. Objects smaller than this are ignored.

maxSize maximum possible object size. Objects larger than this are ignored. If maxSize == minSize model is evaluated on single scale.

Chapter 6
User interface

6 User interface

In this chapter we will talk about GUI (graphical user interface) this is response about the interaction with user to have facility to see the data

Our graphical user interface:

We have implemented 2 graphical interfaces :

1. Flutter GUI

Figure 6.1: Flutter

mobile application that used for:

- Receiving a notification once an abnormal event happened.
- Showing frames of abnormal event occurred.
- Showing Faces that are detected.
- you can select the frame that you want, and you will see the frame and the faces that are relative to this frame.
- you can select the face that you want, and you will see the faces and the frame that is relative to this face.

6.1 What is flutter ?

Flutter is a free and open-source mobile UI framework created by Google and released in May 2017. In a few words, it allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one codebase to create two different apps (for iOS and Android).

- ❖ Flutter consists of two important parts:

- An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).

- A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

6.2 Framework architecture

❖ The major components of Flutter include:

- Dart platform
- Flutter engine (Skia Graphics Engine)
- Foundation library
- Design-specific widgets
- Flutter Development Tools (DevTools)

◄ Dart platform

Flutter apps are written in the Dart language and make use of many of the language's more advanced features. While writing and debugging an application, Flutter runs in the Dart virtual machine, which features a just-in-time execution engine. This allows for fast compilation times as well as "hot reload", with which modifications to source files can be injected into a running application. Flutter extends this further with support for stateful hot reload, where in most cases changes to source code are reflected immediately in the running app without requiring a restart or any loss of state.

For better performance, release versions of Flutter apps on all platforms use ahead-of-time (AOT) compilation.

◄ Flutter engine

Flutter's engine, written primarily in C++, provides low-level rendering support using Google's Skia graphics library. Additionally, it interfaces with platform-specific SDKs such as those provided by Android and iOS. The Flutter Engine is a portable runtime for hosting Flutter applications. It implements Flutter's core libraries, including animation and graphics, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain. Most developers interact with Flutter via the Flutter Framework, which provides a reactive framework and a set of platform, layout, and foundation widgets.

◄ Foundation library

The Foundation library, written in Dart, provides basic classes and functions that are used to construct applications using Flutter, such as APIs to communicate with the engine.

◄ **Design-specific widgets**

The Flutter framework contains two sets of widgets that conform to specific design languages: Material Design widgets implement Google's design language of the same name, and *Cupertino* widgets implement Apple's iOS Human interface guidelines.

6.3 IDE Support

Flutter maintains official support for the following IDEs and editors via plugins:

- IntelliJ IDEA
- Android Studio
- Visual Studio Code

In our Eagle-eye project we have used Android studio to develop a mobile application.

6.4 Widgets

❖ Widgets are generally defined in three basic types:

- Stateful widgets.
- Stateless widgets.
- Inherited widgets.

Being the central class hierarchy in the Flutter framework the three basic types of widgets are used in the construction of every Flutter application. Although all the instances of a widget are immutable, the Stateful widget allows the interaction between user and application. By giving access to the method set State, the state can be maintained in separate state objects. Alternatively, the Stateless widget acts as a constant, and before anything displayed can be changed, the widget has to be recreated. The Inherited widget works by allowing another widget to subscribe to the Inherited widget's state allowing the state to be passed down to its children.

6.5 Why flutter?

- Simple to learn and use

  Flutter is a modern framework, and you can feel it! It’s way simpler to create mobile applications with it. Compared with Java, Swift, or React Native, you'll notice how Flutter is different and simple. Flutter can create a real native application without a bunch of code.

- Quick compilation: maximum productivity

Flutter can change your code and see the results in real-time. It uses Hot-Reload method. It only takes a short amount of time after you save to update the application itself. Significant modifications force you to reload the app. But if you do work like design, for example, and change the size of an element, it’s in real-time!

- Good documentation

  It’s important for new technology to have good documentation. You can learn a lot from Flutter's documentation, and everything is very detailed with easy examples for basic use cases. Each time I’ve had a problem with one of my widgets in my code, I have been able to check the documentation and the answer was there.

- Supported by Android Studio and VS Code

  Flutter is available on different IDEs. The two main code editors for developing with this technology are Android Studio (IntelliJ) and VS Code. Android Studio is a complete software with everything already integrated. You have to download Flutter and Dart plugins to start. VS Code is a lightweight tool, and everything is configurable through plugins from the marketplace.

  We used Android Studio because I don’t need to configure a lot of things to work.

- Design android ,IOS ,desktop and web application in one code

  Flutter allows you to implement a mobile application that works in android apps and IOS and can run the app as desktop ,mobile, web application by using one code.

6.6 Design

- Get started page:

  Once security guard open Eagle-eye Application ,this page will be viewed.

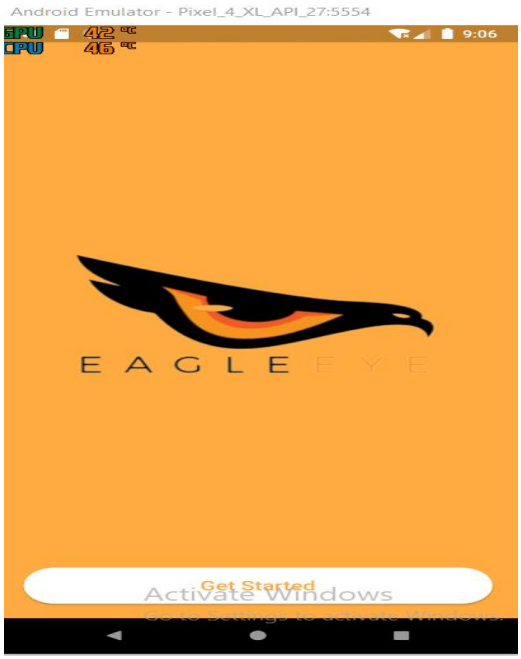


Figure 6.2. Get started page

- Login page:

  Security guard will login by entering email and password.

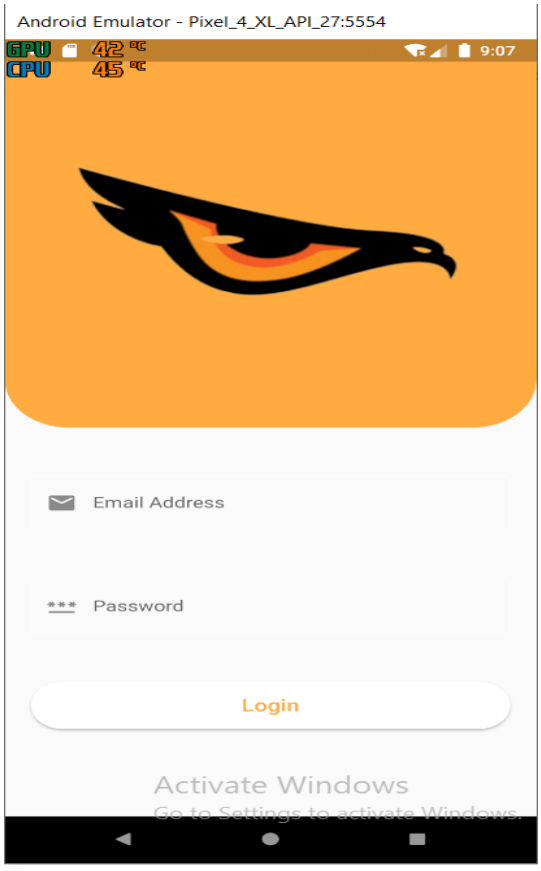


Figure 6.3. Login page

- Show frame page :

  Security guard will be able to show abnormal frames in this page, this page show all abnormal frames detected.

Figure 6.4: Frames page

- Detailed frames page:

  Once a security guard clicks on any frame from frames page, this page will be viewed.

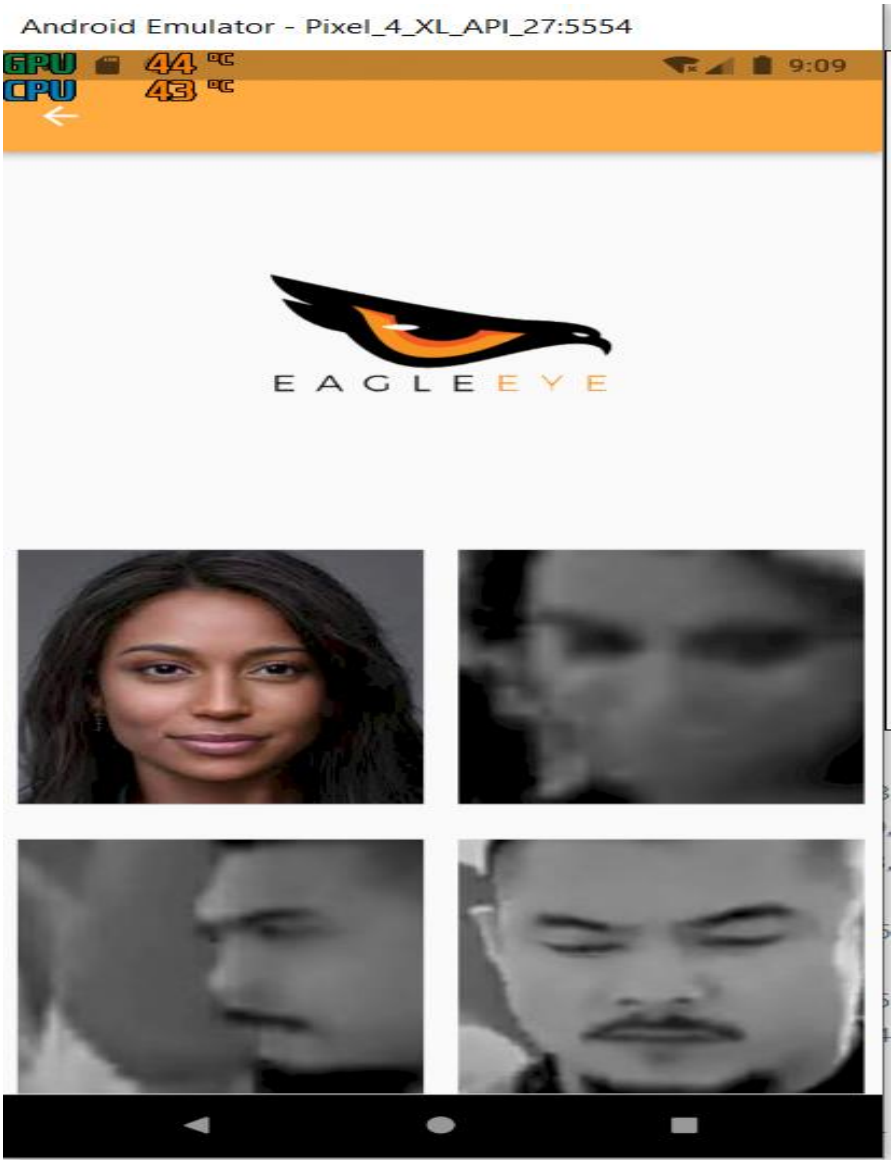


Figure 6.5: ***Detailed frames page***

- Show faces page :

  Security guard will be able to show faces in this page, this page will show all faces that have been detected in abnormal events to be returned whenever security guard needs.

Figure 6.6: Faces page

- Detailed faces page:

  Once security guard clicks on any face, this page will be viewed, this page contains face and frame that the faces has been detected at.

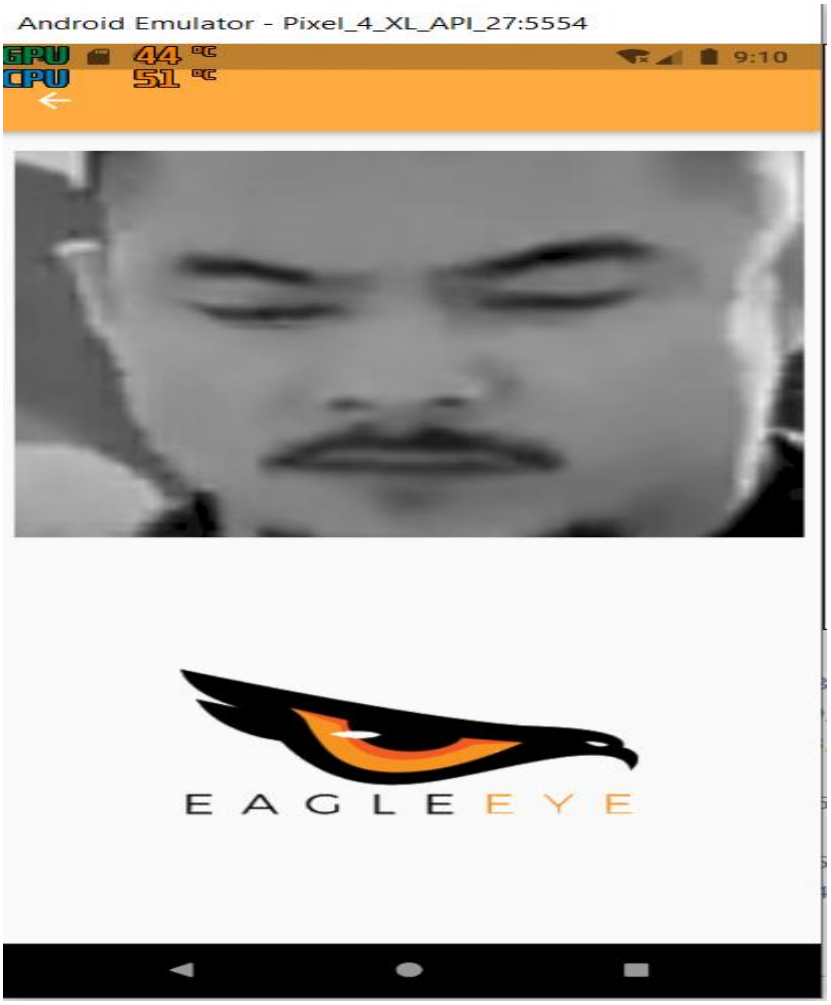


Figure 6.7: Detailed faces page

6.7 Implementation

Sometimes the app crashed without any reason or when we press on the button ,it freeze for five or ten seconds without any response on the screen and all of this problems make the user be frustrating and give bad experience on this app and all of that happened because the code is not organized ,clear, maintainable and testable so to solve this problems we should write clean code that achieve accommodating every feature and scale over time so we will use design pattern bloc.

- **Bloc**

It is the design pattern created by google separate business logic from the presentation layer and enable a developer to reuse code more efficiently.

Bloc is state management library so we should separate the app into some of state so the programmer should know every state the app in, and it helps that every interaction from the user to app should something happened.

Technically for every interaction inside the application, there should be a state emerging from

it. For example, when the data is fetching, the app should be in a loading state displaying a loading animation on the screen. When the internet is off, the application should display a pop-up to let the user know there is no internet connection.

- **Cubic and bloc**

Cubic is the minimal version of the bloc we can say bloc extend cubit; Cubit is a special kind of Stream component based on some functions called from the UI. Functions that rebuild the UI by emitting different states on a Stream.

And we will see example of cubit and state in our app:

The state of app:

```
abstract class frames_state{}
class frames_intialstate extends frames_state{}

class frames_loadingstate extends frames_state{}
class getframes_sucess extends frames_state{}
class getframes_error extends frames_state{
  late final String error;
 getframes_error(this.error);
}
```

Figure 6.8: State of getting Frames

Getting the frames has four state and when the app emits any of this state it allows me to rebuild the screen and know which state am in.

```
 frames_cube() : super(frames_intialstate());

 static frames_cube get(context)=>BlocProvider.of(context);
 FramesData ? framesData ;
 int page=1;
List <Data1>posts=[];

 void get_frames_data (){

   if(state is frames_intialstate ){
   diohelper.getData(path: (frames)).then(
           (value) {
             emit(frames_loadingstate());
             print(value!.data);
             framesData=FramesData.fromJson(value.data);
             posts.addAll(framesData!.data);
             print(posts[0].frameName);

             print(framesData?.data[0].id);
             print('done');

             emit(getframes_sucess());
             page++;
       }
   ).catchError((error){
     print(error.toString());
     emit(getframes_error(error.toString()));
   });
   }
   else{
     String? example = framesData!.links.next;
     var ll=example![example!.length-1];
```

Figure 6.9: cubic of getting frames

```
diohelper.getData(path: ("http://192.168.212.19/api/frames?page="+ll)).then(
        (value) {
          emit(frames_loadingstate());
          print(value!.data);
          framesData=FramesData.fromJson(value.data);
          print(page);

          posts.addAll(framesData!.data);
          print(posts[0].frameName);

          print(framesData?.data[0].id);
          emit(getframes_sucess());
          page++;
      }
  ).catchError((error){
    print(error.toString());
    emit(getframes_error(error.toString()));
  });

 }
}
```

Figure 6.10: cubic of getting frames

from those photos we can see that we separate the logic of UI and places of states.

6.8 Problem faced

In first version we tried to make the desktop with flutter, but we failed because a lot of problem appear:

a. Flutter doesn't have the facilities to record video from the camera on the desktop version of flutter.

b. The desktop version has a lot of problems with the network .

c. we found that it’s hard to communicate with ML model by sending video to it

d. At model, we tried to make the model run at the server, but we found its complicated because sending frames at real time from desktop to the server takes a lot of time and it's complicated and sending video has the same problem

❖ Solution of problem we have faced:

After reading a lot of papers and research, we have decided to implement python GUI to solve all problems that flutter can’t solve.

2. Python GUI

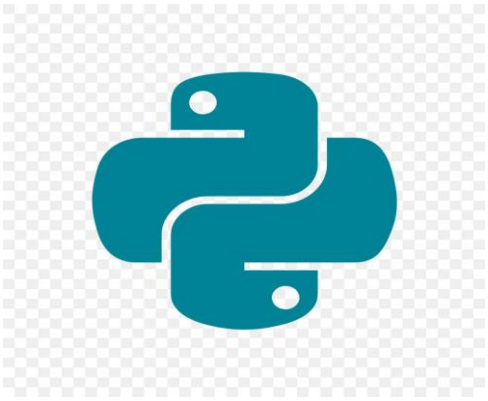


Figure 6.11: Python

Desktop application used for:

- open a camera to record a video
- run machine learning model
- Upload video to check if its normal or abnormal

❖ We have used Tkinter library in python to develop desktop application that can:

- run machine learning model.
- open camera to record video.
- upload a video to detect whether it's normal or not.

6.9 what is Tkinter library?

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

❖ Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps :

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more widgets to the GUI application like button and label .
- putting the main event loop to act against each event occur by the user.

6.10 why python GUI ?

a. it’s easy to link and connect a machine learning model with GUI, so our app can run the model inside it.

b. Tkinter library in python is easy to use and faster to implement.

c. stable and flexible.

d. it’s easy to understand and master.

e. we can use opencv that help us to record video and cutting video into small video.

6.11 Design

- Graphical user interface of desktop application

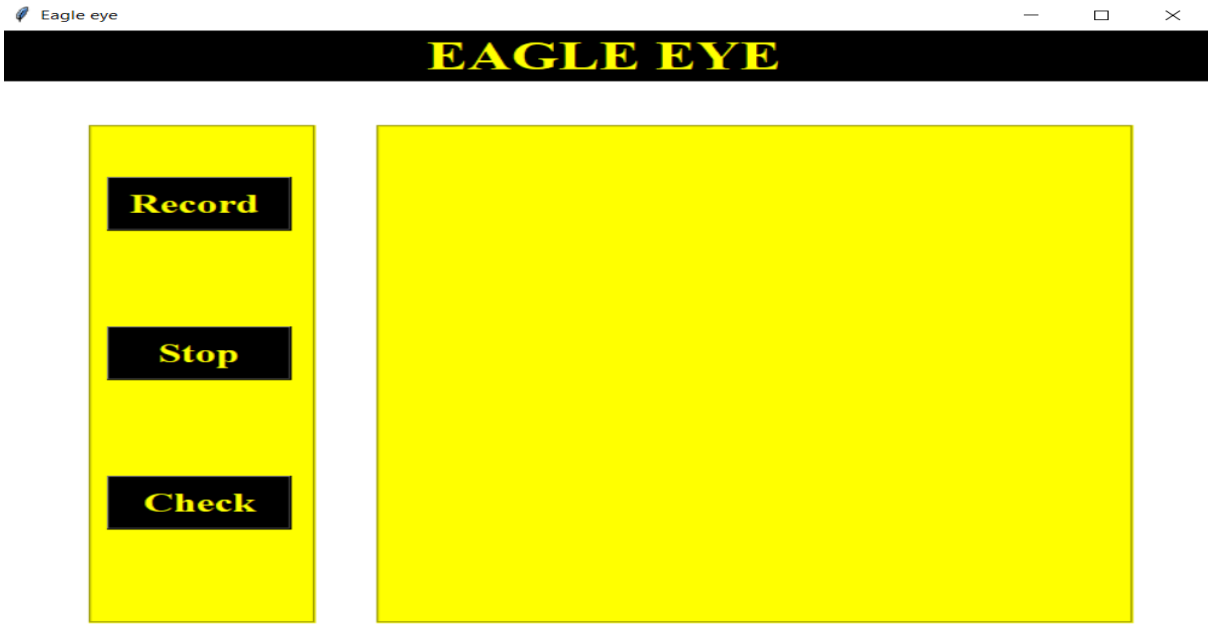


Figure 6.12: Home Page

◄ Record:

Open a camera to record video.

◄ stop:

Stop the recording and send video to ML model to detect anomaly event and show if the video is abnormal or not

◄ check:

It allows the user to upload video from his device to detect anomaly event and show if the video is abnormal or not

- When user clicks detection button and if the video is abnormal will show on screen that its abnormal otherwise it will not show any thing

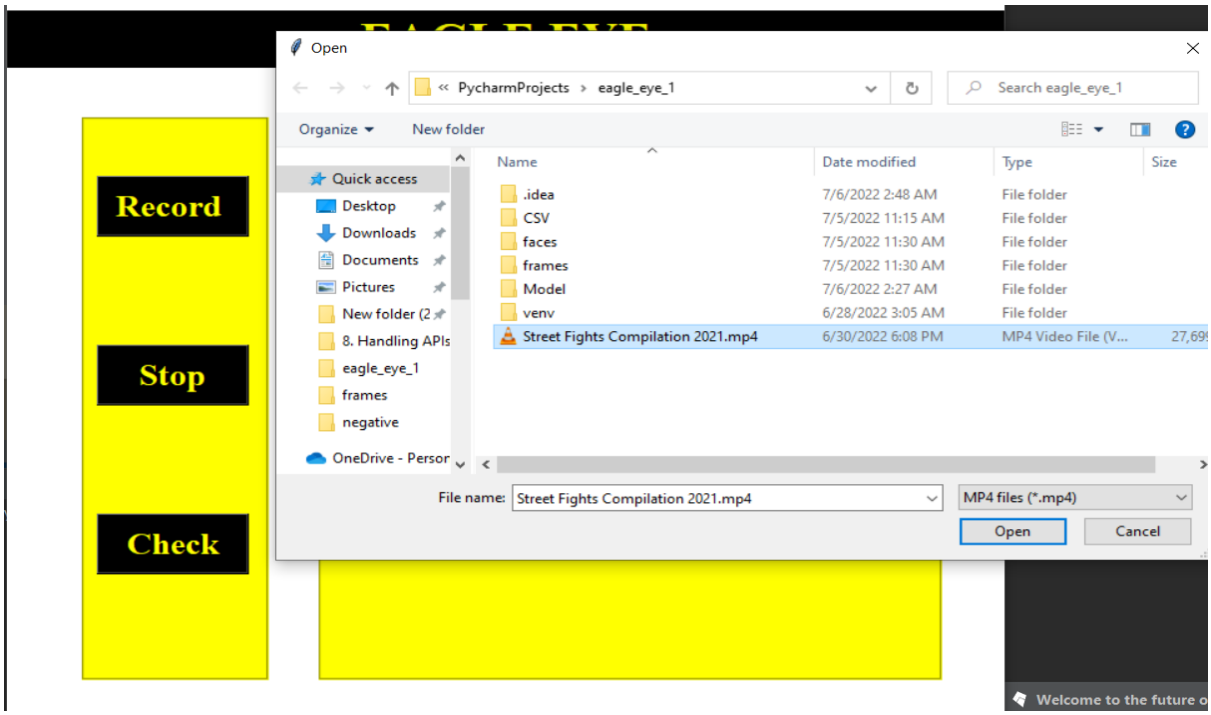


Figure 6.13: Detection page

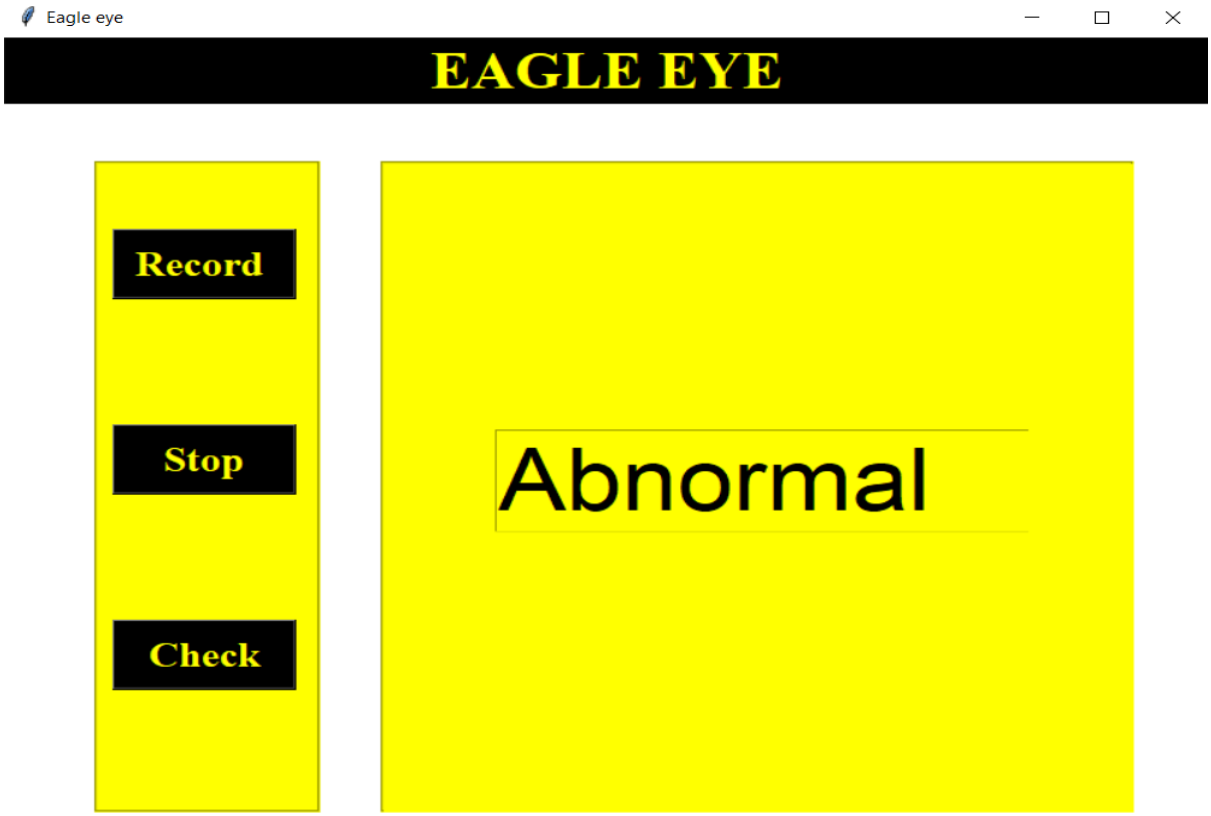


Figure 6.14: output page

Chapter 7
Web Application

7 Web Application

7.1 SW backend

7.1.1 server-side overview

The backend part of any project is responsible for the functionality of the apps, the below figure represents an overview of the whole system.

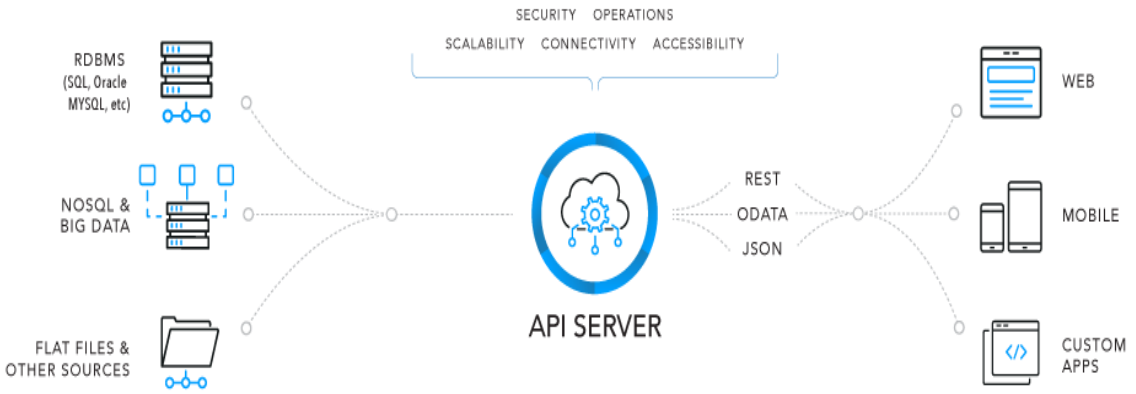


Figure 7.1: backend client-server architecture

Clients can communicate with the server through APIs, as clients could be a mobile application, web application, desktop application ... etc. Backend of the system represents the server-side, such as database architecture and implemented functions; even the backend can produce APIs to facilitate communication between the clients and the server.

Clients of our system are a mobile application which is used by the security guards, and a web application used by the security manager.

Eagle eye system's server-side has two parts of implementation, first part is implemented in Laravel framework and the other in Flask framework.

7.1.2 Server implementation part I

Laravel is a free open-source PHP web application framework that allows you to implement and configure your server-side in php easily and systematically, it also provides the middleware that we will explain later in the APIs section.

Laravel is following the mvc architectural pattern which is an abbreviation of model-view-controller.

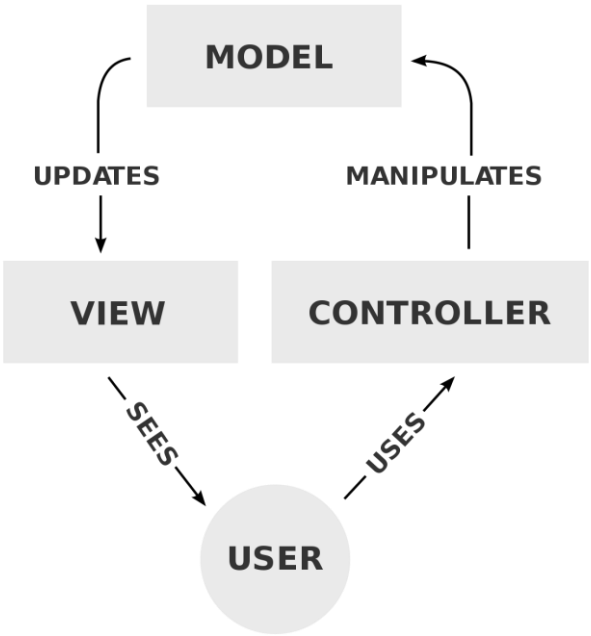


Figure 7.2: model-view-controller architecture

The figure shows the model-view-controller architecture and how they communicate with each other,

MVC is an architectural pattern consisting of three parts:

- Model: Handles data-related logic and communicates directly with the database.
- View: responsible for all the UI logic, as it helps in displaying data fetched from the model to the user in visible interfaces.
- Controller: It controls the data flow into a model object using some implemented functions such as CRUD (create, read, update, delete) and many other needed functions, and updates the view whenever data changes.

7.1.3 Server implementation part II

Eagle eye system's server-side second part of implementation is in Flask.

Flask is a micro web framework written in Python. Microframework here means that it does not require particular tools or libraries, as it is a simple core with simple structure, so it is highly used in building websites and APIs. We used Flask here to implement the APIs that are run and called by the machine learning model and to use the Flask socketio package.

7.1.4 Application Programming Interface (APIs)

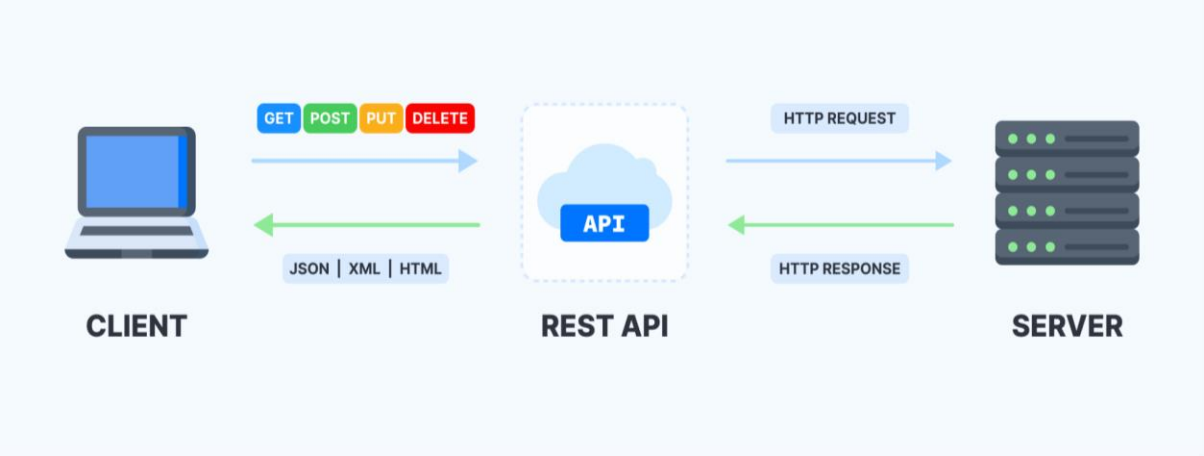


Figure 7.3: REST API Model

As we explained before at backend client-server architecture, Application Programming Interface(API) is the way to communicate and integrate between a software product and another ,clients and the server, for data transmission between them.

7.1.4.1 API implementation

We exposed many APIs for the clients in our server using the REST API model. REST API model ,as shown in the figure, receives a request from the client of any request type ,such as GET, POST, PUT and DELETE, then it sends HTTP request to the server to fetch or manipulate data then sends the response to the client with JSON or XML or HTML format.

Each database entity ,frames and faces, has CRUD operations (create - read - update - delete) to control entity data, it returns JSON response.

Eagle eye system guarantee that when the ML model detects an anomaly event, server sends a notification to the mobile application to alert the security guards, so we used Flask Socketio package to implement the notification API and run it by the ML model

Our system supports multi authentication functionalities, as we mainly have 2 types of actors: security guard and security manager, each actor has different privileges as we explained in more details in the analysis and requirements section. So, we implemented authentication APIs.

7.1.4.2 API testing

1. Frames APIs

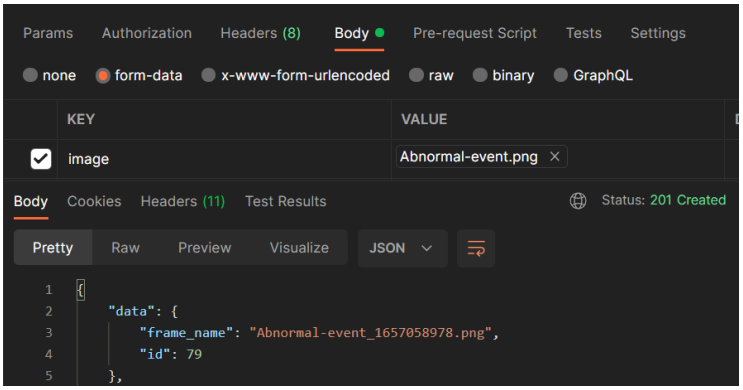


Figure 7.4: Adding a new frame

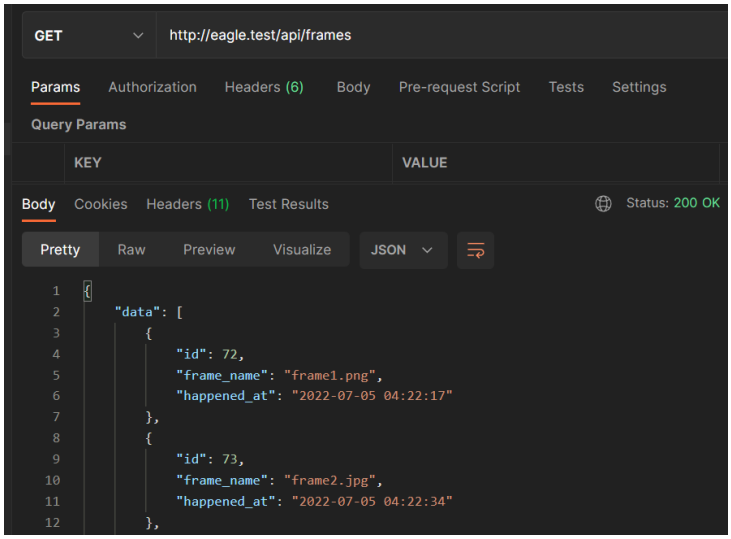


Figure 7.5: Fetching all anomaly frames data

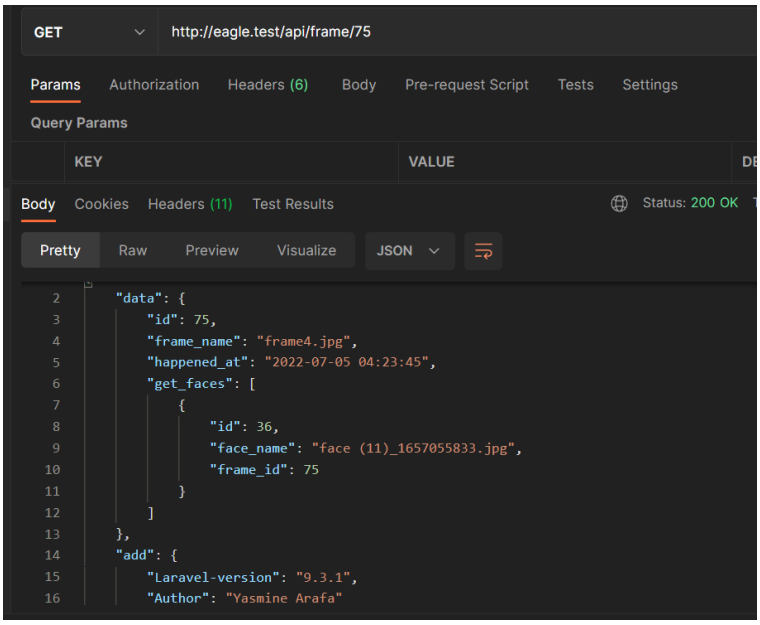


Figure 7.6: find the frame with id = 75 with its related faces data

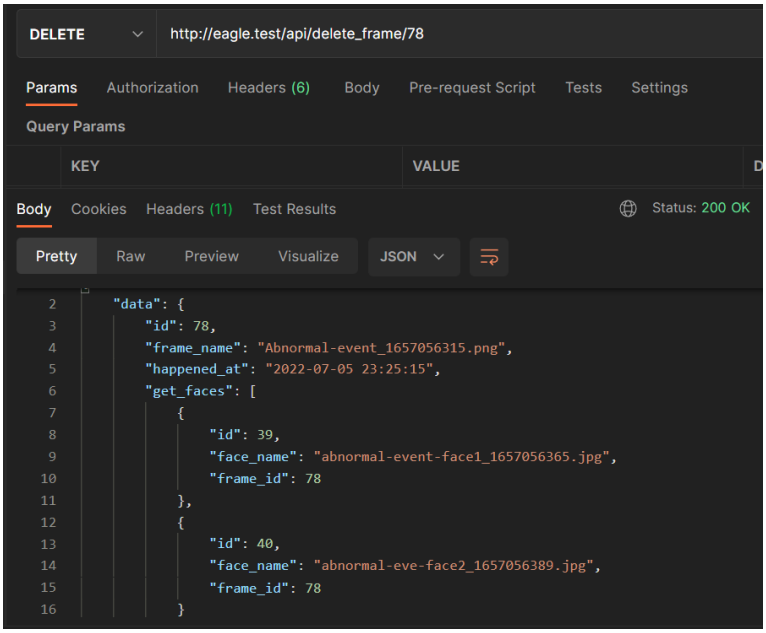


Figure 7.7: delete the frame with id = 78 and related faces

2. Faces CRUD APIs

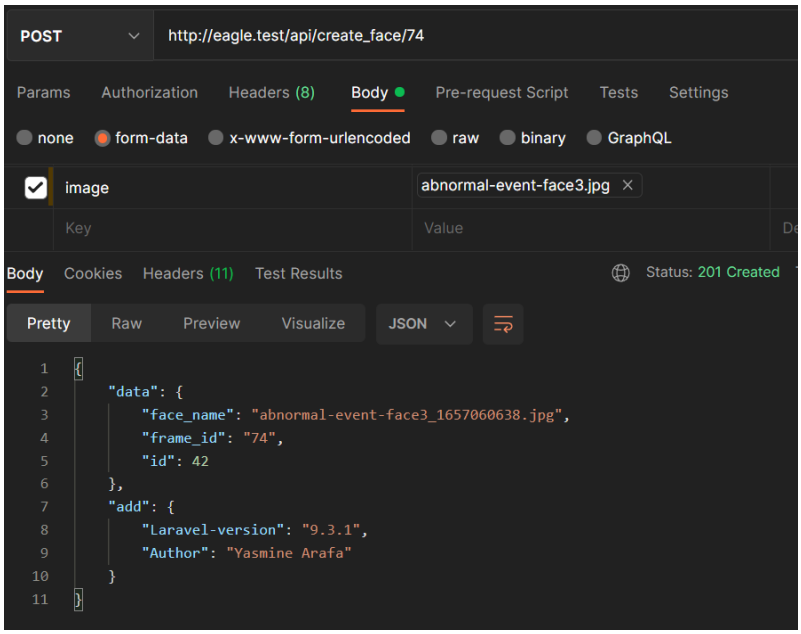


Figure 7.8: adding a face related to the frame with id = 74

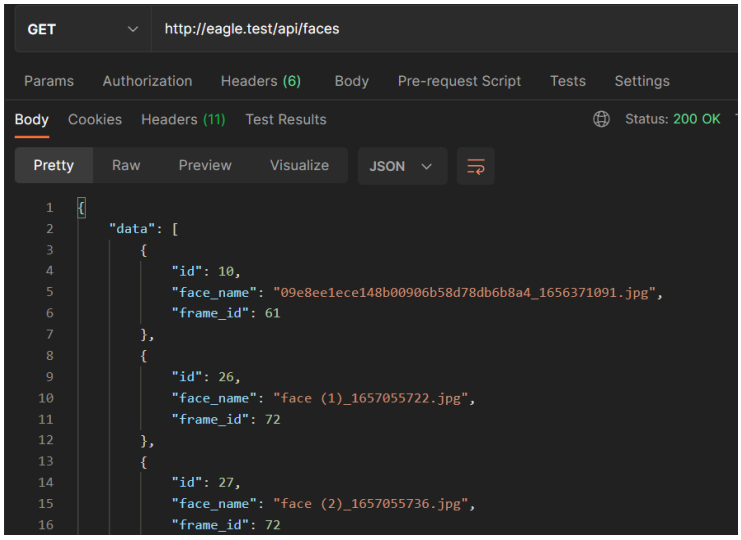


Figure 7.9: Fetching all faces data

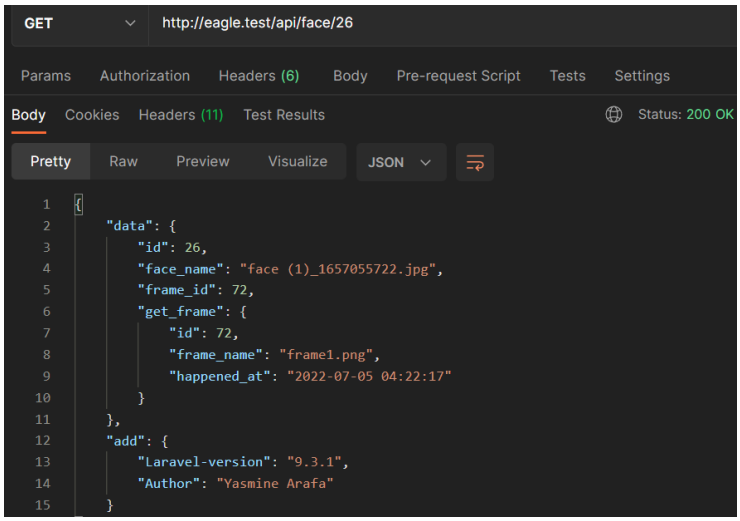


Figure 7.10: find the face with id = 26 with its frame data

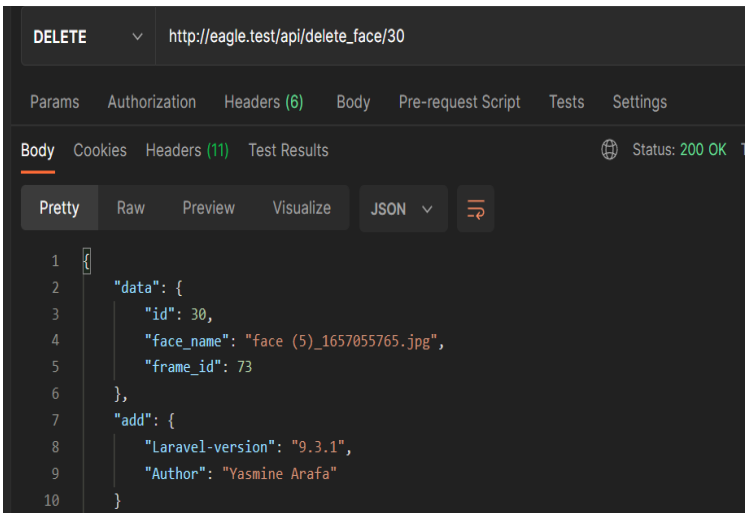


Figure 7.11: delete face with id = 30

3. Socketio notification API

    - sends notification to the mobile application when anomaly event is detected with the first anomaly frame

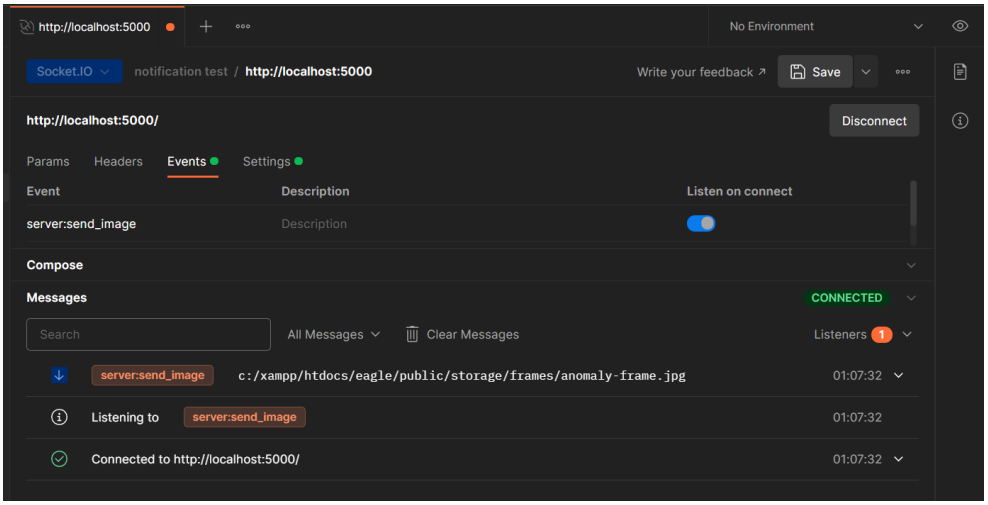


Figure 7.11

The client is listening on the event (server:send_image), Then it receives an anomaly frame path

(c:/xampp/htdocs/eagle/public/storage/frames/anomaly-frame.jpg)

4. User authentication APIs

    - This API is used by the mobile application to verify the users and give them access and privileges

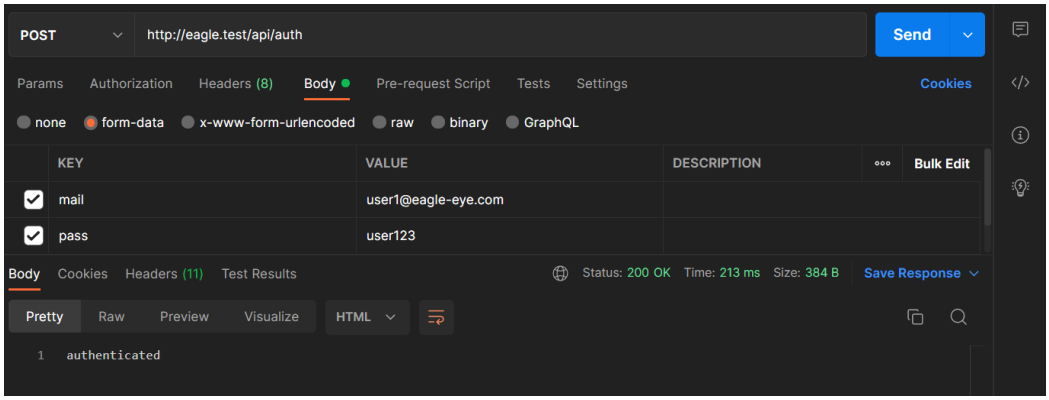


Figure 7.12: Authenticated user

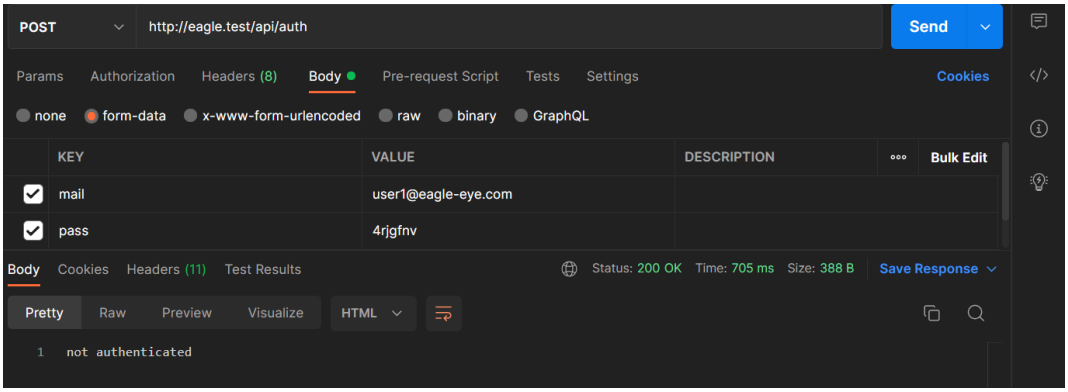


Figure 7.13: Not authenticated user

7.1.5 Task scheduling

As we discussed before in the problems statement, storage is one of the most common problems we have faced in the traditional surveillance system, so we implemented auto deletion functionality using Laravel task scheduling.

7.1.5.1 Laravel Task Scheduling

Laravel's command scheduler provides an approach for managing and running the scheduled tasks on your server-side.

With the scheduler you can define and implement your command in app/Console/Commands then schedule its execution by defining Your task schedule in the kernel (app/Console/Kernel.php) file's schedule method.

```php
class deletion extends Command
{
    /**
     * The name and signature of the console command.
     *
     * @var string
     */
    protected $signature = 'frames:delete';

    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'delete frames older than 30 days';
```

Figure 7.14: command signature and description

```php
/**
 * Execute the console command.
 *
 * @return int
 */
public function handle()
{

    $frames = Frame::where('happened_at', '<',now()->subDays(30)->endOfDay())
    ->get();

    $frameController = new frameController;
    foreach ($frames as $frame) {
        $frameController->destroy($frame->id);
    }

}
```

Figure 7.15: deletion command implementation

Our deletion command handles frames older than 30 days to be deleted automatically for storage saving.

```php
class Kernel extends ConsoleKernel
{

    protected $commands = [
        \APP\Console\Commands\deletion::class,
    ];
```

Figure 7.16: add the deletion command in the kernel

```php
/**
 * Define the application's command schedule.
 *
 * @param  \Illuminate\Console\Scheduling\Schedule  $schedule
 * @return void
 */
protected function schedule(Schedule $schedule)
{
    $schedule->command('frames:delete')->daily();
}
```

Figure 7.17: defining the task schedule in the kernel

System should check the frames older than 30 days every day, so the command is executed daily by the scheduler. To run the scheduler, use the following artisanal command: php artisan schedule:run.

7.1.5.2 Windows Task Scheduler

We used Windows Task Scheduler to automate this command run locally, here are the steps:

1. Create a batch file, example: "auto-deletion.bat"

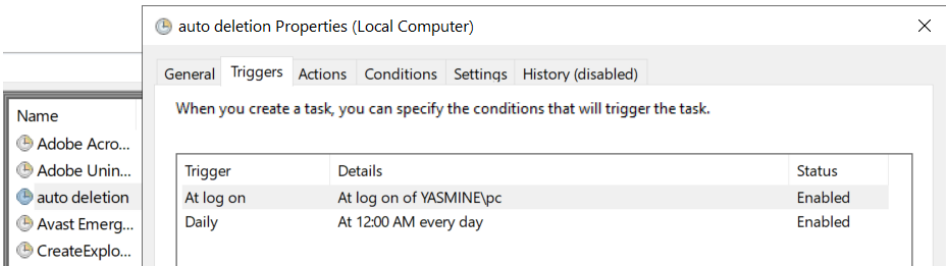
auto-deletion.bat - Notepad

File Edit Format View Help

```
cd C:\xampp\htdocs\eagle\

c:\xampp\php\php.exe artisan schedule:run 1>> NUL 2>&1
```

2. Go to Windows Task Scheduler.
3. Click Create basic task, choose when I logon trigger and then choose Start a program -> your .bat file.
4. Check the open properties dialog option and click Finish.
5. In task properties click Triggers, then click New and add a new trigger repeat task daily.



7.1.6 Database architecture

The PHP Laravel framework is packaged with the Eloquent Object Relational Mapper (ORM), which is a technique used to interact with the database in an Object-Oriented paradigm and provides an extremely easy way to communicate with a database.

As we explained before the Laravel mvc pattern, we use models to deal with databases and handle data. The figures below explain implementation of frame and face models, as the relation between them is one to many so the frame can have many faces.

```php
class Frame extends Model
{
    protected $table = 'frames';
    public $primarykey = 'id';
    public $timestamps = false ;
    use HasFactory;

    public function getFaces(){
        return $this->hasMany('App\Models\Face');
    }
}
```

Figure 7.18: Frame model

```php
class Face extends Model
{
    protected $table = 'faces';
    public $primarykey = 'id';
    public $timestamps = false ;
    use HasFactory;

    public function getFrame(){
        return $this->belongsTo('App\Models\Frame', 'frame_id');
    }
}
```

Figure 7.19: Face model

ER diagram is an entity–relationship model describes and specifies relationships that can exist between entities with each other.

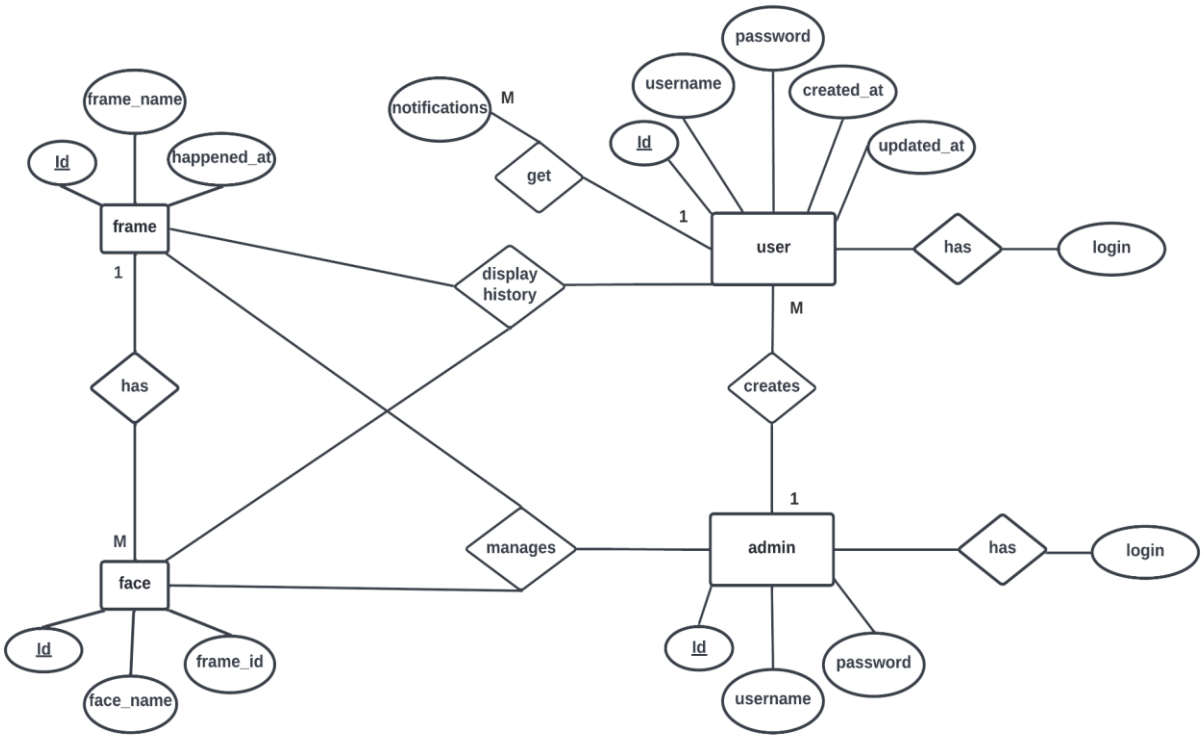


Figure 7.20: ER diagram

7.2 Frontend

7.2.1 Tools and packages

- HTML

  HTML stands for Hyper Text Markup Language which is the standard markup language for creating web pages. HTML is used to describe the structure of a web page and its syntax depending on series of elements in form of tags <>, these elements form the structure of the web page content

- CSS

  CSS ,stands for Cascading Style Sheets, describes the style of the HTML elements and how they are displayed in the webpage. CSS can be written in 3 ways (external - internal - inline), external means using CSS as an external stylesheet.

- Laravel Blade Templates

  Blade is the simple and powerful templating engine that is included with Laravel. Blade allows you to use plain PHP code in your templates, actually all Blade templates are compiled into plain PHP code. Blade template files use the .blade.php file extension and are typically stored in the resources/views directory.

Blade views may be returned from routes or controllers using the global view helper, we can pass data to the Blade view using the view helper's second argument, for example:

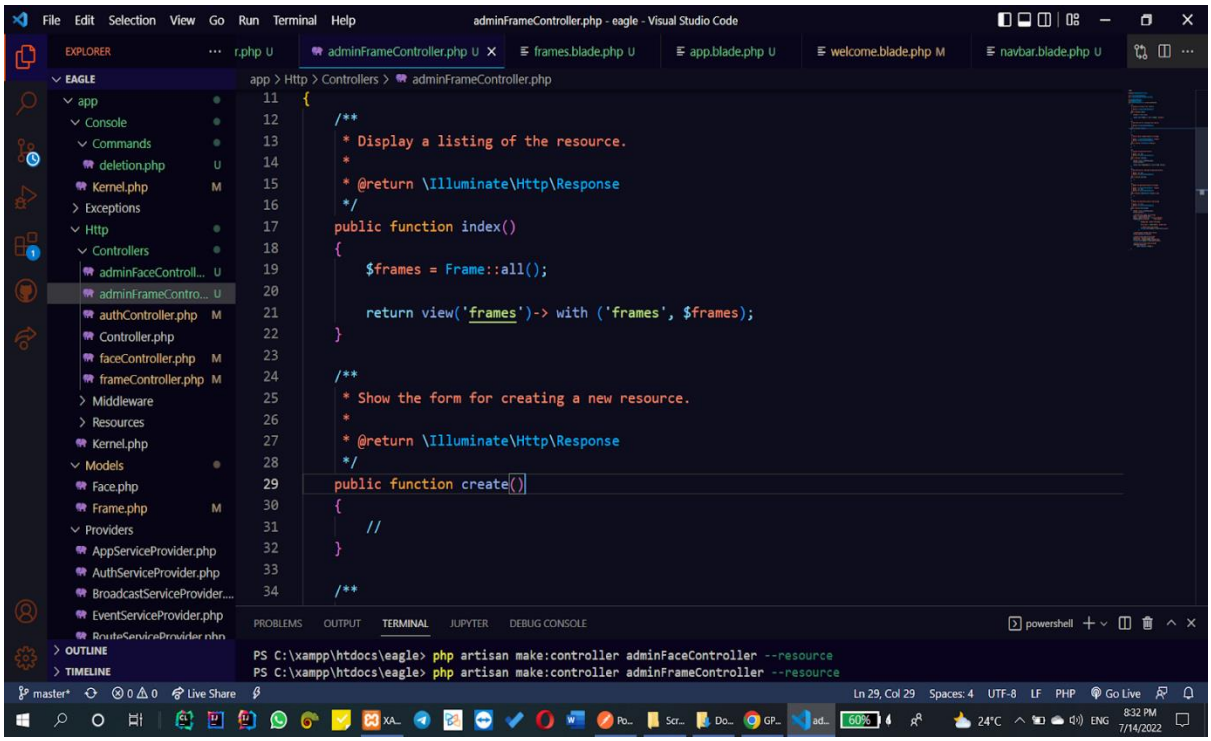


Figure 7.21: frame Controller index file

Index function returns all frames, then passes it to “frames” view.

- Laravel collective

  The Laravel Collective package HTML comes packed with an HTML and FORM generator allowing you to handle easy to manage forms in your blade files as well as intricate model binding to your forms.

Installation:

```
$ composer require laravelcollective/html
```

Using Laravel collective form, for example:

Figure 7.22

This form deletes any frame by passing its id to the frameController' destroy function.

7.2.2 Dashboard interface

The dashboard allows the admin to display all frames, faces, delete any frame and add a new guard (user).

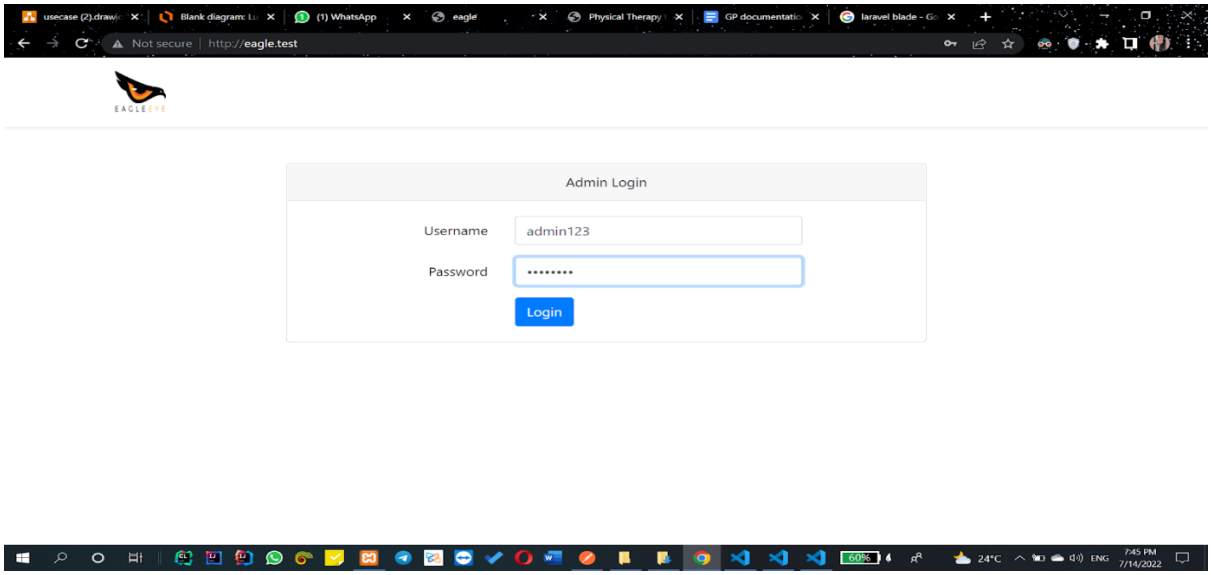


Figure 7.22: Admin login view in the dashboard

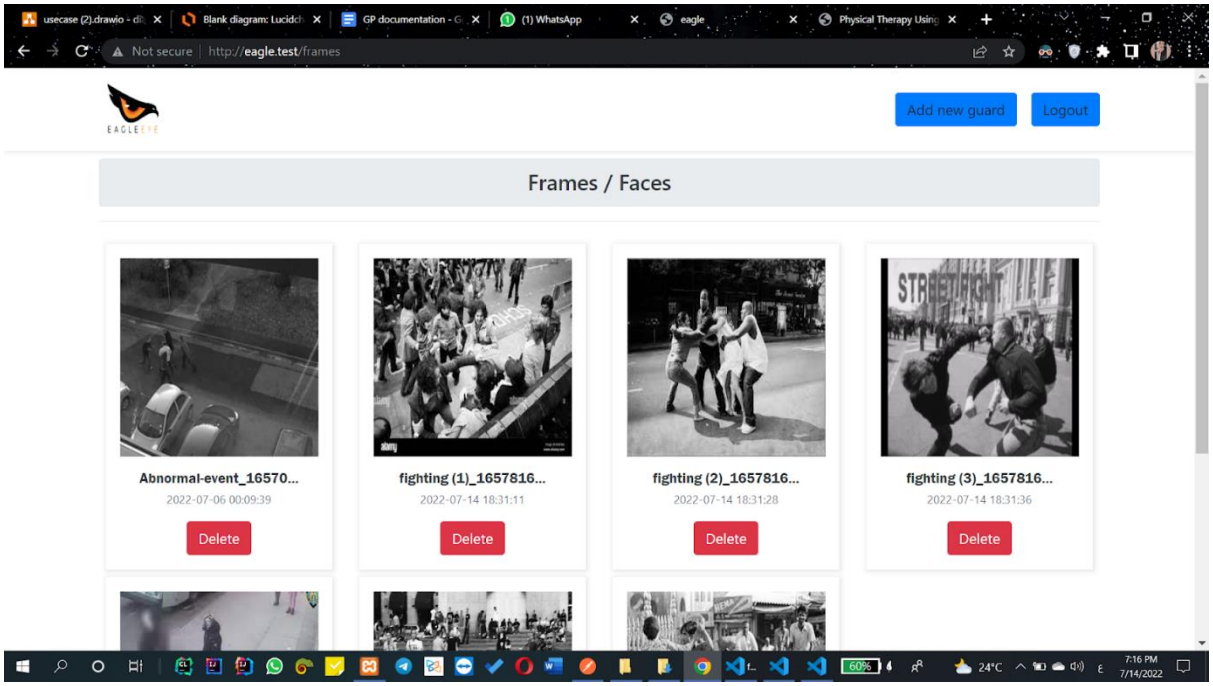


Figure 7.23: Frames view in the dashboard

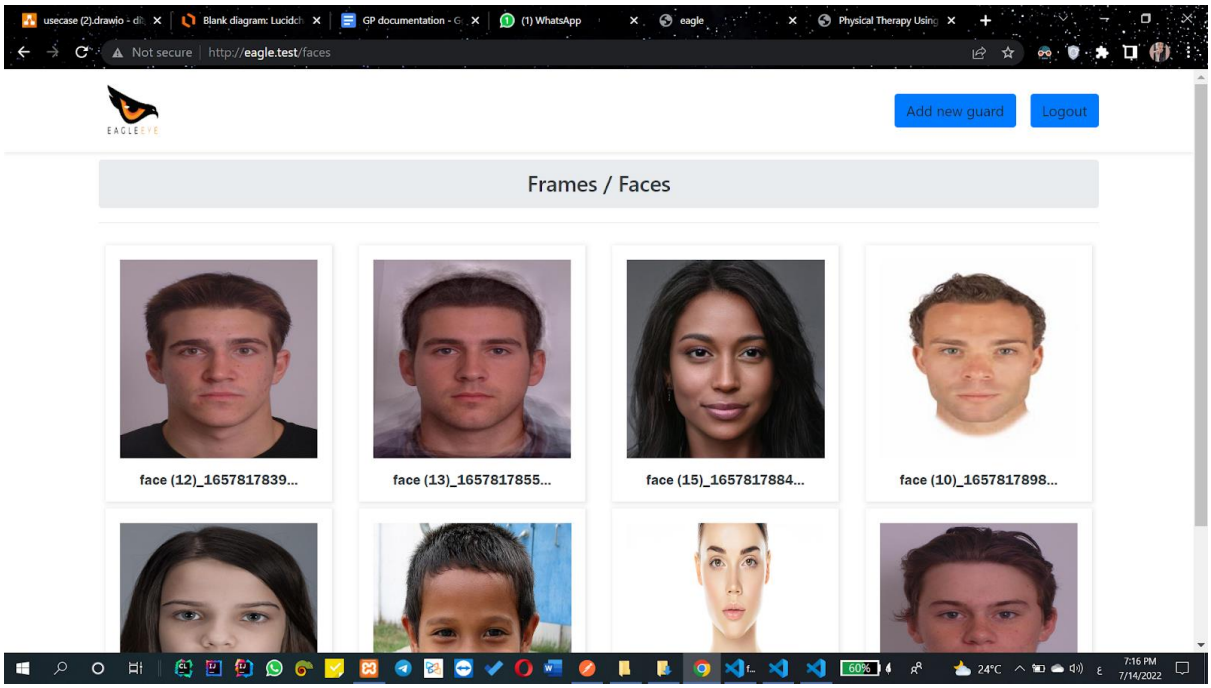


Figure 7.24: Faces view in the dashboard

Chapter 8
Conclusion and future work

8 Conclusion and future work

8.1 Conclusion

Because of a lot of crimes that happens and sometimes be hard to know when happened and where and who make the problem and who effected with this problem so We want to make surveillance system that has the ability to be fast and accurate and give the user good experience on the our system so we make supervised model that detect fight and has this abilities to detect very fast and accurate called SVM and to make good services to the user we provide 2 GUI one desktop and another is mobile application desktop will be for security monitor and the mobile application will be for the guard and it will have the facility to show abnormal frames and faces and get notification when abnormal event happen so we will save time and we can rescue the people faster and we made fast backend that can serves our application at Realtime so the user feel good and the backend provide API that insert frame and faces and show frames ,show faces ,show selected frame and show selected face and for admin we make dash board that give the admin full control from add user and delete frames and show frames and show faces and in our system we have auto deletion that after 30 days the system will remove old frames to get free space and save money.

8.2 future work

The Eagle-Eye project is a prototype system until now, we implemented and developed it in a reliable, Scalable process which allows us to add features, enhance and maintain the system.

Our plan for future of the Eagle-Eye

1. Replacing the recording live video feature of the desktop with real stablished Ip camera
2. For desktop we will make livestream 24 hours and security monitor will have the ability to select which cam he wants and the accuracy percentage.

Chapter 9
References

9 References

[1] EarthWeb. 2022. *Who Has the Most CCTV in The World in 2022? (New Stats)*. [online] Available at: <https://earthweb.com/who-has-the-most-cctv-in-the-world/>

[2] ReportLinker. (2022, March 22). *Surveillance Technology Global Market Report 2022*. GlobeNewswire News Room. Retrieved from https://www.globenewswire.com/news-release/2022/03/22/2407975/0/en/Surveillance-Technology-Global-Market-Report-2022.html

[3] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey, 2019.

[4] Santhosh Kelathodi Kumaran, D. P. Dogra, and P. Roy. Anomaly detection in road traffic using visual surveillance: A survey. ArXiv,abs/1901.08292, 2019.

[5] 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.

[6] D. Xu, E. Ricci, Y. Yan, J. Song, and N. Sebe. Learning deep representations of appearance and motion for anomalous event detection. In BMVC, 2015.

[7]] S. Wu, B. E. Moore, and M. Shah. Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. In CVPR, 2010.

[8] S. Wu, B. E. Moore, and M. Shah. Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. In CVPR, 2010.

[9] Gyuwan Kim, Hayoon Yi, Jangho Lee, Y. Paek, and S. Yoon. Lstm-based system-call language modelling and robust ensemble method for designing host-based intrusion detection systems. ArXiv, abs/1611.01726, 2016.

[10] Runtian Zhang and Qian Zou. Time series prediction and anomaly detection of light curves using lstm neural network. Journal of Physics: Conference Series, 1061:012012, 07 2018.

[11] Zhengying Chen, Yonghong Tian, Wei Zeng, and Tiejun Huang. Detecting abnormal behaviours in surveillance videos based on fuzzy clustering and multiple auto-encoders. 2015 IEEE International Conference on Multimedia and Expo (ICME), pages 1–6, 2015.

[12] 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.

[13] 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.

[14] Raghavendra Chalapathy, A. Menon, and S. Chawla. Anomaly detection using one-class neural networks. ArXiv, abs/1802.06360, 2018.

[15] Sovan Biswas and R. Venkatesh Babu. Short local trajectory based moving anomaly detection. Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing, ICVGIP '14, New York, NY, USA, 2014. Association for Computing Machinery.

[16] Y. Gao, H. Liu, X. Sun, C. Wang, and Y. Liu. Violence detection using oriented violent flows. Image and Vision Computing, 2016.

[17] J. Kooij, M. Liem, J. Krijnders, T. Andringa, and D. Gavrila. Multi-modal human aggression detection. Computer Vision and Image Understanding, 2016.

[18] Datta A, Shah M, Lobo NDV. Person-on-person violence detection in video data. In: Pattern Recognition, 2002. Proceedings. 16th International Conference on. vol. 1. IEEE; 2002. p. 433–438.

[19] S. Mohammadi, A. Perina, H. Kiani, and M. Vittorio. Angry crowds: Detecting violent events in videos. In ECCV, 2016.

[20] X. Cui, Q. Liu, M. Gao, and D. N. Metaxas. Abnormal detection using interaction energy potentials. In CVPR, 2011.

[21] I. Saleemi, K. Shafique, and M. Shah. Probabilistic modeling of scene dynamics for applications in visual surveillance.TPAMI, 31(8):1472–1485, 2009

[22] C. Lu, J. Shi, and J. Jia. Abnormal event detection at 150 fps in matlab. In ICCV, 2013

[23] B. Zhao, L. Fei-Fei, and E. P. Xing. Online detection of unusual events in videos via dynamic sparse coding. In CVPR, 2011.

[24] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In CVPR, 2014

[25] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In ICCV, 2015

[26] D Chen, P Wang, L Yue, Y Zhang, and T Jia. Anomaly detection in surveillance video based on bidirectional prediction. Image and Vision Computing, 98:103915, 04 2020

[27] F. Li, W. Yang, and Q. Liao. An efficient anomaly detection approach in surveillance video based on oriented gmm. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1981–1985, 2016.

[28] N. Li, F. Chang, and C. Liu. Spatial-temporal cascade autoencoder for video anomaly detection in crowded scenes. IEEE Transactions on Multimedia, pages 1–1, 2020.

[29] D. Ryan, S. Denman, C. Fookes, and S. Sridharan. Textures of optical flow for real-time anomaly detection in crowds. In 2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pages 230–235, 2011.

[30] Jing Wang and Zhijie Xu. Spatio-temporal texture modelling for real-time crowd anomaly detection. Comput. Vis. Image Underst., 144(C):177–187, March 2016.

[31] Y. Yuan, J. Fang, and Q. Wang. Online anomaly detection in crowd scenes via structure analysis. IEEE Transactions on Cybernetics, 45(3):548–561, 2015.

[32] J. Zhong, N. Li, W. Kong, S. Liu, T. H. Li, and G. Li. Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1237–1246, 2019.

[33] Yi Zhu and Shawn Newsam. Motion-aware feature for improved video anomaly detection. 07 2019.

[34] 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.

[35] 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

[36] Marco Bertini, Alberto Del Bimbo, and Lorenzo Seidenari. Multiscale and real-time non-parametric approach for anomaly detection and localization. Computer Vision and Image Understanding, 116:320–329, 03 2012.

[26+10] https://www.sciencedirect.com/topics/computer-science/client-server-architecture

[27] https://laravel.com/docs/9.x/

[28] https://flask.palletsprojects.com/en/2.1.x/

[29] https://pypi.org/project/Flask-SocketIO/

[30] https://laravel.com/docs/9.x/scheduling

[31] https://laravel.com/docs/9.x/eloquent

[32] Flutter: BLoC vs Cubit. ¿Should I use BLoC or Cubit? 🤔 This… | by Pablo Reyes | Medium