WLAN Edge Computer

Jayamali D.C
Uduwela U.D.P.D
Premabandhu V.R.K.M
Madurapperuma M.A.D.T.L

Supervised by

Dr. Sampath Edirisinghe

June 25, 2023

Department of Computer Engineering

University of Sri Jayewardenepur

Acknowledgements

Foremost, we would like to express our sincere gratitude to our final year research supervisor, Dr. Sampath Edirisinghe of the Department of Computer Engineering, the University of Sri Jayewardenepura for their enthusiasm, patience, insightful comments, and proper guidance that have helped us tremendously at all times in this research project.

We also extend gratitude and appreciation to our lecturers in the Department of Computer Engineering for allowing us to do the research and for the guidance. We are also grateful to all the personnel who helped us with the completion of this project.

Contents

List of Figures

List of Acronyms

FOE	-	Faculty of Engineering
USJ	-	University of Sri Jayewardenepura
LAN	-	Local Area Network
WLAN	-	Wireless Local Area Network
LSTM	-	Long Short-Term Memory
RNN	-	Recurrent Neural Network

Abstract

Edge computing enables the processing of data close to the network edge. Hence, edge computing provides low latency, reduces the data volume transported in the networks, reduces the cost of operation, and increases the privacy of data. A WLAN edge computer adds computational and storage capabilities to household or industrial WLANs. Thereby, services such as surveillance video processing, IoT data processing, and industrial control can operate with minimum latency.

This project would involve a design of a WLAN edge computer that directly connects to a Wi-Fi router. The devices in the network will connect to the WLAN edge computer, where server applications would process the data of these applications. The objective of this project is to implement a WLAN Edge Computer that can be used in a home environment with the features, Surveillance video processing, human recognition, pose detection, smart home application.

Chapter 1

Introduction

Wireless local area networks (WLANs) and edge computing are two separate technologies that are often used together to improve the performance and efficiency of networked systems. WLANs are wireless networks that use radio waves to provide wireless high-speed Internet and network connections. They can be used to connect devices such as laptops, smartphones, and tablets to the Internet, as well as to connect devices within a local area network (LAN). Edge computing is a distributed computing paradigm that brings computation and data storage closer to the sources of data, such as sensors, cameras, and other devices. This allows for faster processing and analysis of data, as well as reduced latency and bandwidth requirements. LAN edge computing can be used to improve the performance and efficiency of surveillance video processing systems. One example of this is using edge devices such as gateways or cameras with built-in edge computing capabilities, to perform tasks such as video compression, object detection, and facial recognition. This can help to reduce the amount of data that needs to be transmitted over the network to a central location for further processing, and also reduce the amount of data stored and processed in the cloud. Another example is using edge computing to perform real-time analytics on the surveillance video, such as identifying and tracking objects, detecting suspicious behavior, and raising alarms. This can help to improve the effectiveness of the surveillance system, and also reduce the amount of data stored and processed in the cloud. Overall, WLAN edge computing can be used to improve the performance and efficiency of

surveillance video processing systems by performing video analysis and analytics at the edge of the network, close to the cameras and other sensors, reducing the amount of data that needs to be transmitted over the network, as well as reducing latency and bandwidth requirements. WLAN edge computing also can be used to improve the performance and efficiency of smart home applications. The edge devices can collect data from various sensors, such as temperature and motion sensors, and perform tasks such as controlling

lights, thermostats, and security cameras.

1.1 Background

With the proliferation of surveillance systems and the increasing adoption of smart home technologies, the need for efficient data processing and analysis has become paramount. Traditional approaches to video processing and smart home automation often rely on centralized cloud-based architectures, leading to challenges such as high latency, increased network congestion, and potential privacy concerns. To overcome these limitations, WLAN edge computing has emerged as a promising solution. WLAN edge computing leverages the computational capabilities of edge devices, such as access points and routers, to perform data processing and analysis closer to the source. By bringing intelligence to the edge, WLAN edge computing aims to enhance the performance and efficiency of surveillance video processing and smart home applications. This approach enables real-time decision-making, reduces the reliance on cloud infrastructure, and addresses the limitations associated with latency and bandwidth constraints. As a result, WLAN edge computing has gained significant attention as a means to improve the performance and responsiveness of surveillance systems and smart home environments.

1.2 Objectives

The objectives of utilizing WLAN edge computing for surveillance video processing and smart home applications to improve performance are as follows:

- Reduce Latency: The primary objective is to minimize latency in surveillance video processing and smart home applications. By performing data processing and analysis at the edge, closer to the data source, the aim is to achieve real-time or near-real-time decision-making, ensuring timely response and action.
- Improve System Efficiency: WLAN edge computing aims to enhance the overall system efficiency by offloading computation-intensive tasks from centralized cloud infrastructure to edge devices. This objective includes optimizing resource utilization, reducing network congestion, and improving the scalability and reliability of the system.

- Enhance Privacy and Security: Edge computing enables data processing and analysis to be performed locally, reducing the need for transmitting sensitive surveillance video or smart home data to external servers. The objective is to enhance privacy and security by minimizing the exposure of data to potential risks associated with cloud-based solutions.
- Enable Smart Home Automation: WLAN edge computing aims to support intelligent automation in smart homes. The objective is to enable efficient data processing and analysis for tasks such as home security, energy management, and device control, thereby enhancing the functionality and convenience of smart home environments.
- Optimize Network Bandwidth: By leveraging edge computing capabilities, WLAN edge computing aims to optimize network bandwidth usage. This objective involves reducing the amount of data transferred over the network by performing data filtering, compression, and local analysis at the edge, leading to reduced bandwidth requirements.
- Enable Edge Intelligence: The objective is to leverage the computational capabilities of edge devices to enable advanced analytics, machine learning, and artificial intelligence algorithms at the edge. This objective empowers surveillance systems and smart home applications to make intelligent, context-aware decisions without relying solely on cloud-based processing.

1.3 Initial Milestones

1. Literature Survey
2. Planning the best approach for the project
3. Implement the video surveillance models
4. Implement smart home application
5. Designing and planning the architecture of WLAN
6. Fine tuning and testing the system as a whole

7. Latency comparison
8. Finalization

Chapter 2

Literature Review

Wireless Local Area Network (WLAN) edge computing has gained significant attention in recent years due to its potential to enhance performance and efficiency in various applications. This literature review focuses on the utilization of WLAN edge computing in surveillance video processing and smart home applications to improve performance. By leveraging the computational capabilities of edge devices, such as access points and routers, data processing and analysis can be performed closer to the source, reducing latency, network congestion, and enhancing overall system efficiency.

1. Edge Computing in Surveillance Video Processing:

Viani, F., & Zorzi, M. (2018). Smart cameras for edge video analytics: Perspectives and challenges. IEEE Internet of Things Journal, 5(6), 4463-4480.

This paper discusses the integration of edge intelligence in smart cameras for real-time video analytics. It explores the benefits of edge computing in surveillance applications, including reduced latency and bandwidth requirements.

Saha, S., Roy, S., & Mukherjee, S. (2020). Video analytics in IoT-enabled smart surveillance system: A comprehensive review. IEEE Internet of Things Journal, 7(3), 1989-2011.

This comprehensive review presents an overview of video analytics techniques in IoT-enabled smart surveillance systems. It highlights the importance of edge computing in processing and analyzing surveillance video data for real-time decision-making.

2. WLAN Edge Computing for Smart Home Applications:

Zhang, Y., Mao, Y., Leng, S., & Hu, F. (2020). A survey on fog computing for the Internet of Things. IEEE Internet of Things Journal, 7(7), 6329-6350.

his survey paper provides an overview of fog computing (a related concept to edge computing) for the Internet of Things (IoT). It discusses the role of fog computing in smart home applications, including real-time data processing and resource management.

Alippi, C., & Serranti, S. (2020). Fog and edge computing for enhanced smart living environments. IEEE Internet of Things Journal, 7(10), 9023-9031.

The authors discuss the integration of fog and edge computing paradigms to enhance smart living environments, including smart homes. It presents various use cases and applications of fog/edge computing in the context of smart home automation and energy management.

3. Performance Improvement in WLAN Edge Computing:

Li, Z., Liu, Y., Li, Z., Xu, W., & Zhang, Y. (2019). A low-latency edge computing architecture for IoT surveillance systems. IEEE Internet of Things Journal, 6(4), 6754-6765.

This paper proposes a low-latency edge computing architecture specifically designed for IoT surveillance systems. It addresses the challenges of real-time video processing by offloading computation-intensive tasks to edge devices, resulting in reduced latency and improved performance.

Kim, D. H., & Jeong, Y. S. (2020). Performance evaluation of an edge computing-based smart home system using WLAN and IoT devices. Sensors, 20(11), 3042.

The authors evaluate the performance of an edge computing-based smart home system that utilizes WLAN and IoT devices. The study investigates the impact of edge computing on latency, response time, and energy consumption, showcasing the benefits of this approach

Chapter 3

Methodology

3.1 Video Surveillance application

3.1.1 Implementation of Video Surveillance Camera System

Video Surveillance system is one of the applications that has been implemented within the WLAN Edge Computer prototype. In this application, video frames collected from CC TV cameras are sent to the Edge Computer for identifying and alerting threats. Because of the major goal of this project is for utilizing the Wireless LAN of the target premises, it was required to design and implement the Video Surveillance Camera setup to get the use of WLAN.

Simple demonstration of the designed system can be indicated as below.

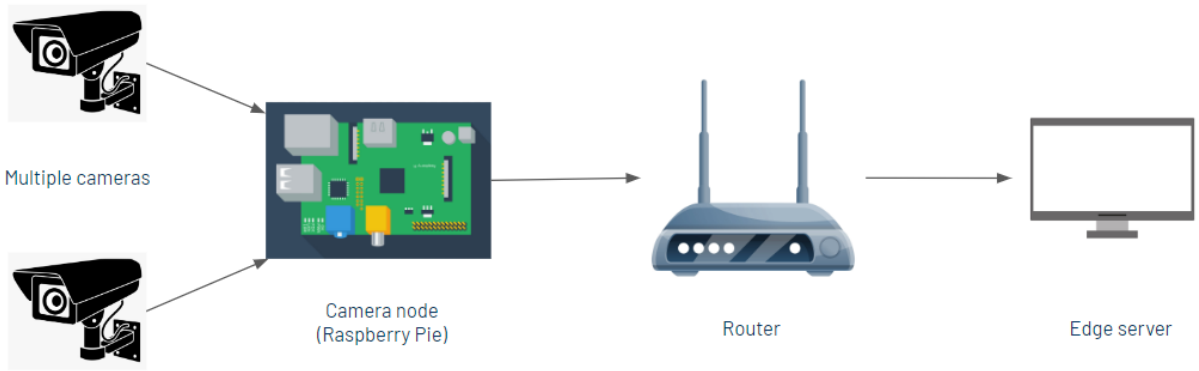

Figure 1 : System design

A camera node was needed for coordinating the cameras in the system. This camera node is capable of transmitting the video frames collected from each camera to the Edge Computer. Overall architecture of the system that derived from the above skeleton can be shown as below.

Overall Architecture of the System

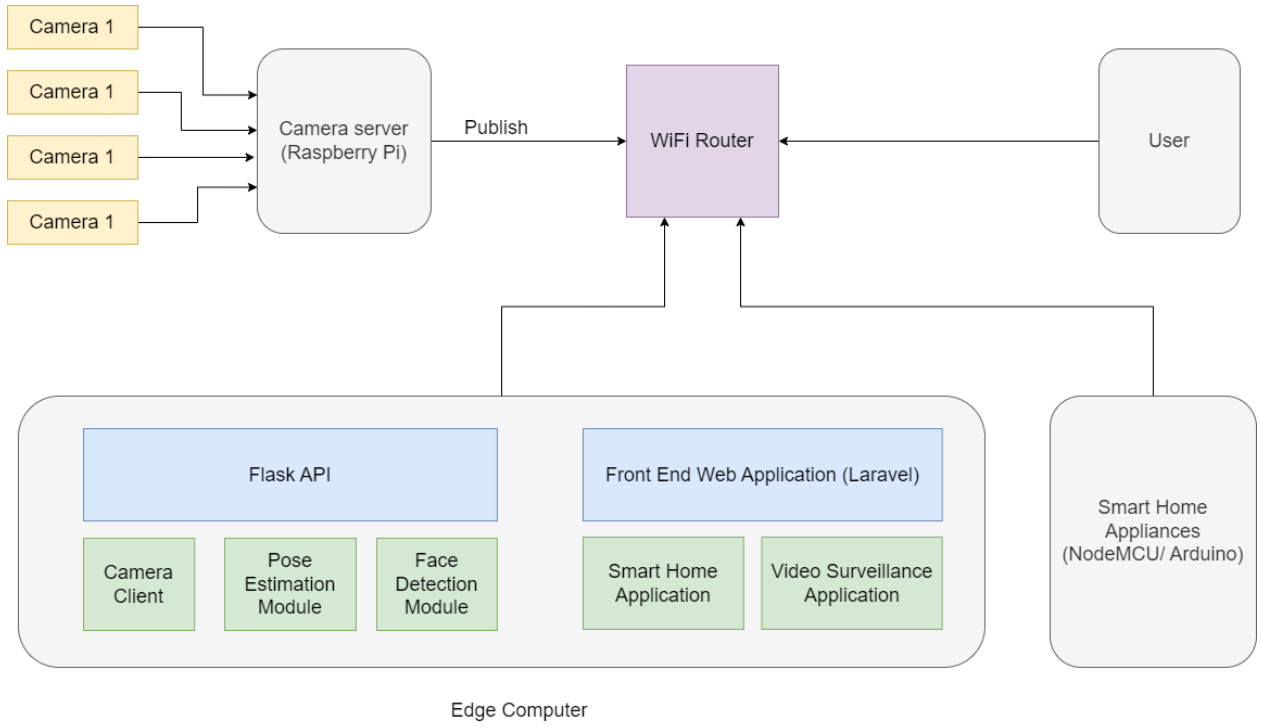

Figure 2 : Overall architecture

Camera Server

Camera server is responsible for collecting the video frames from each camera and sending them to the edge server. For implementing the camera server python Vidgear library is used.

Vidgear Library

The VidGear library is a powerful and flexible Python library for video capturing and processing. It provides an easy-to-use interface for capturing frames from various video sources, such as webcams, IP cameras, screen recordings, video files, and more. VidGear also offers a wide range of features for real-time video processing, including frame manipulation, color space conversions, video stabilization, object detection, and more.

Once the camera server is started, it publishes the video frames from the each camera to the wireless LAN. The message header is included with a port number, so the client application in the LAN can receive the video frames from the corresponding port.

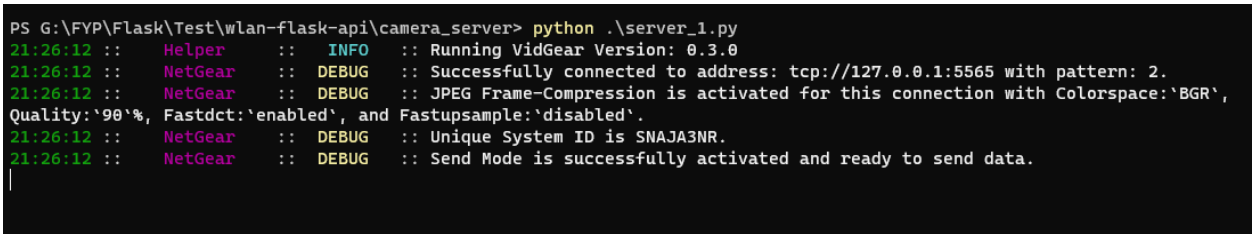

Figure 3 : A camera server running

Flask API for Video Processing

Flask API is consist with three main modules. They are,

1. Camera Client

   When a request arrived from the frontend for video frames of a camera, then the video frames from the corresponding port are send to the frontend as the response.

2. Pose Estimation Module

   Processing a single frame costs considerable time. So each and every frame cannot be processed for pose estimation, because it may cause laggings in frontend application video feed. So as a solution, the program was designed to process 1 frame for 300 frames. If the frame is going to be processed, then a separate thread is created for continuing the pose estimation. Once the results are written to the database thread will be destroyed.

3. Face Detection Module

   Face recognition process does not cost much time and hardware like the pose estimation module. But for reducing the hardware usage and ensuring the smooth video streaming for the frontend 1 frame for 50 frames are set to be processed.

```python
def gen_frames(cam_num):

    client = clients[cam_num]

    i = 0
    # loop over
    while True:

        # receive frames from network
        frame = client.recv()

        # check for received frame if Nonetype
        if frame is None:
            # frame = frame_offline
            break

        # {do something with the frame here}

        ################################################################################
        if(i == 0):
            threading.Thread(target=estimate_poses, args=(frame))

        if(i % 20 == 0):
            frame = detect_faces(frame)
            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
        ################################################################################

        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
                b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')  # concat frame one by one and show result
        i = (i + 1) % 300

    client.close()
```

Figure 4 : Module integration for frame analysis

Challenges

```python
clients = []
i = 0
if len(clients) == 0:
    while(i < 4):

        port_num = 5565 + i

        for proc in process_iter():
            for conns in proc.connections(kind='inet'):
                if conns.laddr[1] == port_num:
                    proc.send_signal(SIGTERM)
                    continue
        # define various tweak flags
        options = {"flag": 0, "copy": False, "track": False}

        # Define Netgear Client at given IP address and define parameters
        client = NetGear(
            address="127.0.0.1",
            port=str(port_num),
            protocol="tcp",
            pattern=2,
            receive_mode=True,
            logging=True,
            **options
        )

        clients.append(client)
        print(5565 + i)
        i = i + 1
```

Figure 5 : Mechanism for reuse camera client

The major challenge was found when integrating the camera module with the flask

API. Because the flask API was unable to re-use the address (port) of the camera once it is used. So alternative mechanism was needed to be implemented to kill the processes listening on the relevant port once it is needed.

3.1.2 Pose Classification and threatful pose detection

Development of a Surveillance Video Application with Threatful Pose Detection

The increasing need for advanced surveillance systems to ensure public safety has led to the development of pose detection models capable of identifying threatening actions. This report outlines the development of a surveillance video application that aims to detect threatful poses in real-time, enhancing the efficiency of surveillance operations.

Methodology

Data Collection

To obtain a diverse dataset of threatening poses, a web scraping tool called Fatkun was utilized. This tool facilitated the collection of images from various online sources, representing poses such as climbing, running, pushing, pulling, and gun shooting. These images were crucial for training and validating the threatful pose detection model.

Keypoint Extraction

The MediaPipe framework was employed for keypoint extraction from the collected images. MediaPipe's pose estimation module utilizes advanced algorithms to detect and extract key body keypoints, including wrists, elbows, knees, and shoulders. By capturing the spatial positions of these keypoints, the system gains the ability to recognize specific threatening poses accurately.

Mediapipe provides a robust solution capable of predicting **thirty-three 3D landmarks** on a human body in real-time with high accuracy even on CPU. It utilizes a two-step machine learning pipeline, by using a detector it first localizes the person within the frame and then uses the pose landmarks detector to predict the landmarks within the region of interest.

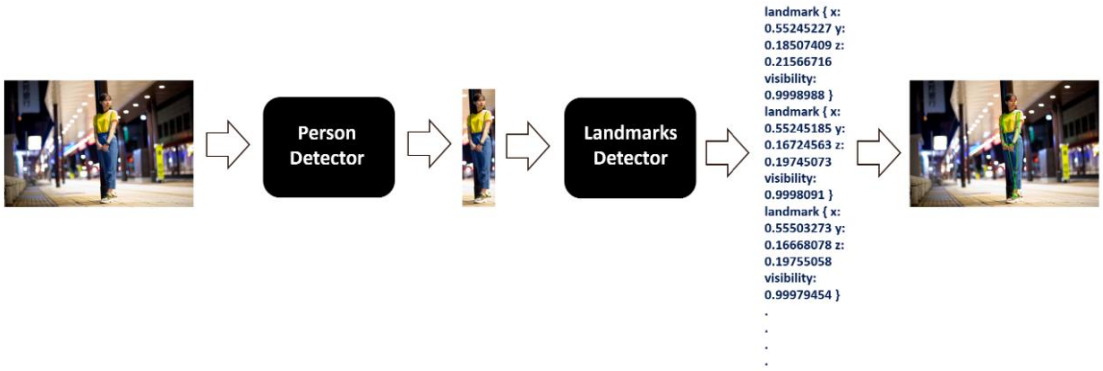

Figure 6: Key points detection using mediapipe

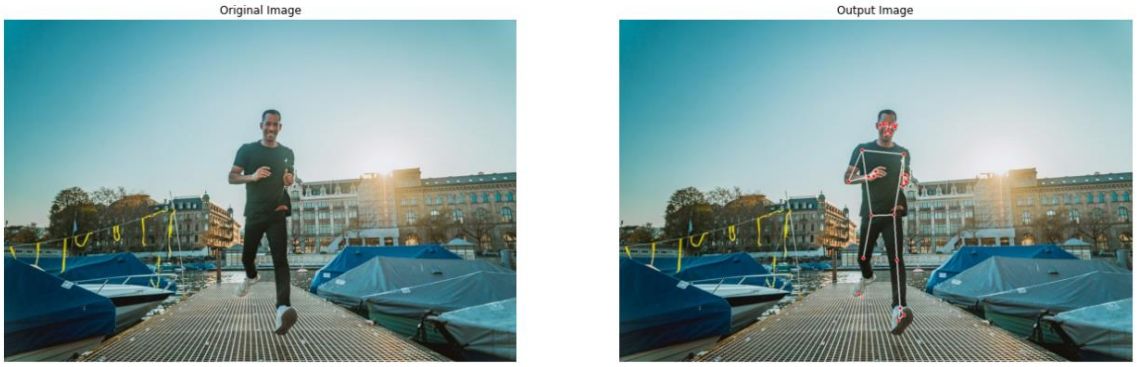

Figure 7: Sample skeleton

Data Preprocessing

The extracted keypoints from MediaPipe were saved in a structured format, such as a CSV file. Before training the KNN classifier, the data underwent preprocessing steps, including cleaning the dataset, handling missing values, and normalizing the keypoints. These steps were essential to ensure accurate and reliable classification results.

Pose Classification

The preprocessed keypoints were utilized to train a K-nearest neighbors (KNN) classifier. KNN is a simple yet effective algorithm that assigns a class label to a new pose

based on the majority vote of its nearest neighbors in the feature space. The number of neighbors and other hyperparameters were fine-tuned using techniques such as cross-validation to optimize the classification accuracy.

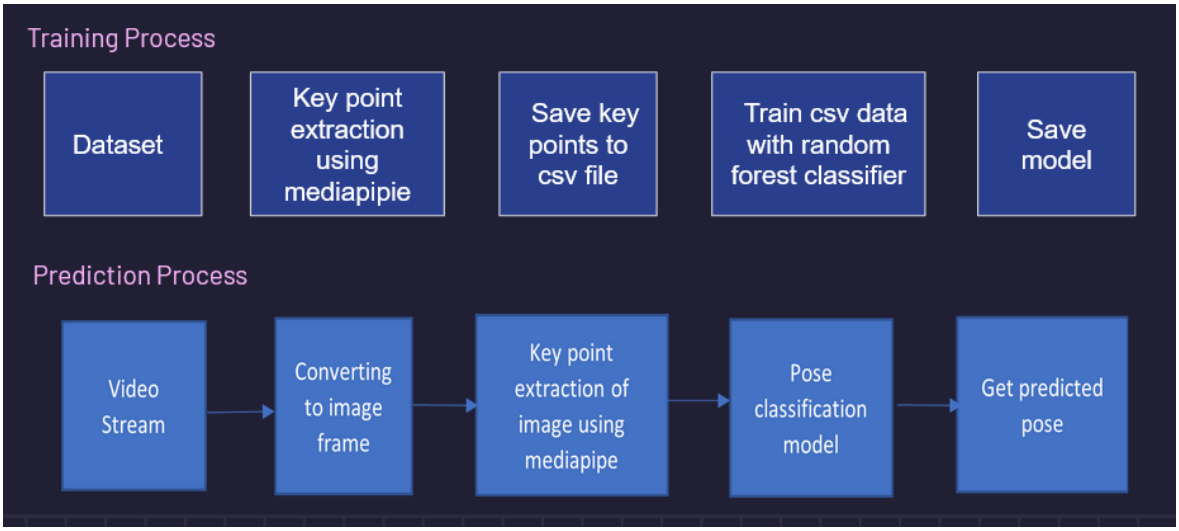

Figure 8: Training and prediction process

To evaluate the performance of the developed surveillance video application, several metrics were employed. Precision, recall, and F1-score were calculated to measure the accuracy of pose classification. A separate test dataset, distinct from the training data, was used to ensure unbiased evaluation and assess the model's generalization capabilities.

The results demonstrated promising accuracy in detecting and classifying threatening poses. The application achieved high precision, recall, and F1-score for each pose category, indicating its effectiveness in identifying potential threats within surveillance videos. The system also demonstrated robustness when tested with real-world video scenarios, further validating its reliability.

3.1.2 Face recognition

At the beginning of the project, face_recognition library in python was used to implement the face recognition module. It is a famous package used for face detection, face recognition and face manipulation tasks.

This method was chosen for the implementation due to some reasons, and ease of use was one of them. Face_recognition library provides a simple and intuitive API which makes it easy to integrate face recognition capabilities into python applications. Since integrating the face recognition module with other modules and applications of the project was a major concern of the project, this advantage was considered to be very important at the beginning of the project. Also, the library providing both face detection and face recognition features was another reason that was considered when choosing this method. The library offers robust face detection capabilities. It can locate faces in both images and videos and return the coordinates of the bounding box around each face which can be used for further processing. And when it comes to face recognition, it compares and recognizes faces, accurately. Another reason as to why this method was selected was that this library comes with pre-trained models that can be readily used for face detection and recognition which eliminates the necessity for training our own models from scratch. This was considered as a great help in saving time. Also, performance was mainly considered when choosing this method. This library was designed for efficient processing which allowed real-time face recognition tasks. This facilitated parallel processing and optimizations to achieve high performance even when processing multiple images or video frames simultaneously which was very beneficial for our project.

However, after implementing the module we faced two unexpected challenges even though the module detected and recognized faces, accurately and efficiently. One of them was the model not being able to recognize multiple faces. Since multiple face recognition was a main requirement of the project, this was considered as a main drawback of this model. Also, the disability of the model to recognize faces that are few meters away from the camera was another drawback of the model because if that requirement is not satisfied, it adds no value to the objective of the project. Therefore, research was done on how to overcome this challenges.

Research was carried out to find a better approach to implement the face recognition module which will be able to overcome the identified drawbacks of the implemented model while maintaining its beneits. After researching on various methods and techniques, the final decision was made on haarcascade classifier and YOLO. Performance and efficiency are crucial factors of our project. When it comes to those factors, Haar Cascade cassifier shows better results than YOLO. Also, it can reliably detect

and recognize faces in controlled environments with consistent lighting conditions. Since we have the control over the placement of the final hardware product of our project and the lighting of the surroundings, it is not a problem to maintain necessary lighting conditions. Also, YOLO has a higher computational cost compared to Haar Cascade cassifier, specially on resource-constrained devices. Training a YOLO model requires a significant amount data and computational power, and fine-tuning a pre-trained YOLO model for specific face detection tasks can be challenging because it requires substantial expertise and access to relevant datasets. Therefore, after considering all these factors Haar Cascade classifier was chosen to implement the model.

Implemented module was able to overcome the weaknesses that the previously implemented model had and performed accurately and efficiently. And the implemented face recognition module performed 3 functionalities;

1. Face Detection
2. Face Registration
3. Face Recognition

1. Face Detection

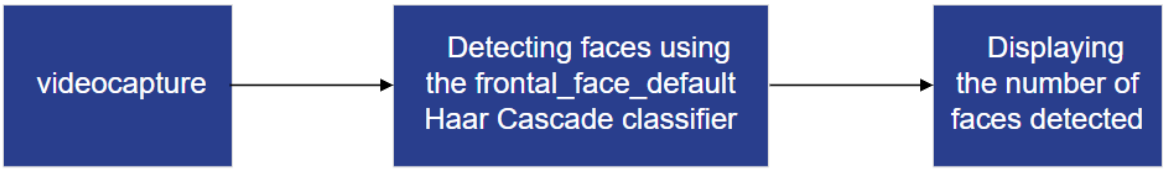

Figure 9: Face detection process

Five pre-trained cascade XML files were used for the detection process and they are

- Haarcascade-frontal-face
- Haarcascade-eye
- Haarcascade-eye-tree-eyeglasses
- Haarcascade-smile
- Haarcascde-fullbody

The classifier was initialized first by loading the pre-trained XML files. And once the classifier is initialized and a video stream is applied, the classifier scans through each

frame of the video at multiple positions and scales, and searches for regions that matches the learnt features. And it detects the object as a face, if a match is found. The implemented model counts the number of faces detected in a frame and shows the count in the application.

2. Face Registration

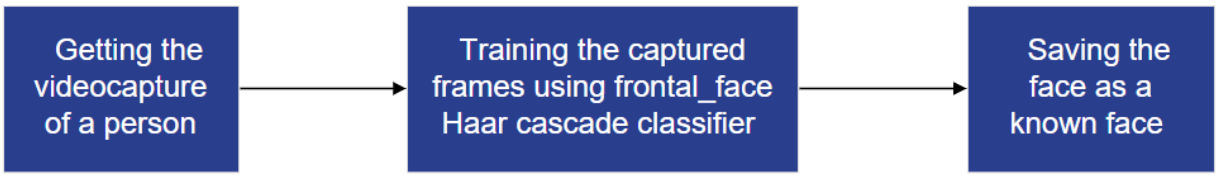

Figure 10: Face registration process

When it comes to building face registration functionality, haarcascade-frontal face classifier was used. Model was registered with multiple faces to increase the accuracy of the model. To ensure that the model registers a face accurately, each person needed to be present in front of the camera for around 100s. This can be considered as a drawback of the model, however, since the registration needs to take place only once, the impact of this to efficiency can be considered to be negligible.

3. Face Recognition

Figure 11: Face recognition

Face recognition was done using haarcascade-frontal face classifier, as well. In this process if the detected face was recognized as an already registered person, the corresponding name was displayed on the application. If not it was displayed as 'Unknown'.

Database

After implementing the model, it was connected to a MySQL database to store the records of the model. A table named ‘records’ was created and maintained in the database to record each person detected and the corresponding date and time the person was detected. Time was used as the primary key of the table when recording data. And then a connection was established between the database and the laravel project which was the frontend application of the entire project to display the real-time face recognition data in the frontend dashboard. This data was displayed through an implemented table in the laravel project.

Chapter 4

Experiments and Results

4.1 Surveillance Video application- Pose Detection

The developed surveillance video application, incorporating a threatful pose detection model, has achieved a commendable accuracy rate of 86% when tested with a set of evaluation images. The application successfully detects and classifies threatening poses such as climbing, running, pushing, pulling, and gun shooting. The model's accuracy demonstrates its efficacy in accurately identifying these poses and contributes to enhancing public safety.

The model's accuracy was assessed using a separate set of testing images, distinct from the training data. Evaluation metrics, including precision, recall, and F1-score, were employed to measure the model's performance comprehensively. By achieving an accuracy of 86%, the model has demonstrated its ability to correctly classify a significant majority of poses within the defined threatful categories.

Figure 12: Pose prediction results

Latency Analysis:

The surveillance application was hosted on both an edge server and the Heroku cloud server to assess latency performance. Latency refers to the time it takes for the application to respond to a request. By hosting the model on an edge server, which is closer to the source of the data, the application aims to minimize latency and achieve faster response times compared to a remote cloud server like Heroku.

Analysis of the application's latency on the edge server and Heroku cloud server provides valuable insights into their respective performance. The lower latency observed on the edge server suggests that processing the pose detection locally, closer to the source of the video feed, results in faster response times. This reduced latency can be crucial for real-time applications and ensures more immediate threat detection and response.

Cloud (Heroku)	Edge
3.245677s	2.889977s
4.556533s	3.778009s
3.134457s	2.009932s
2.789334s	1.899303s

Heroku Link- `https://surveilancecharukaj.herokuapp.com/predict`

Figure 13: Pose prediction results

4.2 Surveillance Video application - Face Recognition

1. Face Detection

The implemented model detects the object as a face, if a match is found. And then counts the number of faces detected in a frame and shows the count in the application.

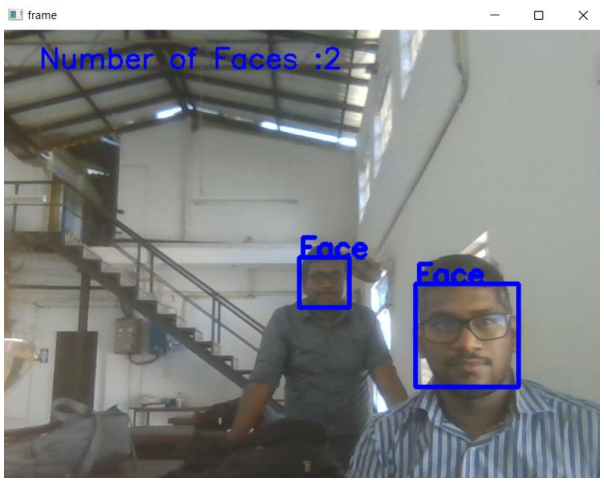

Figure 14: Face detection results

2. Face Registration

Implemented model takes around 100s to register a face, accurately. Time can be decreased but it will affect the accuracy of the model. It is upto the user to decide the amount of time that he or she is willing to spend on registration of a face. The more the time spent the better the accuracy will be. Time taken is displayed on the application until the user decides to quit the registration.

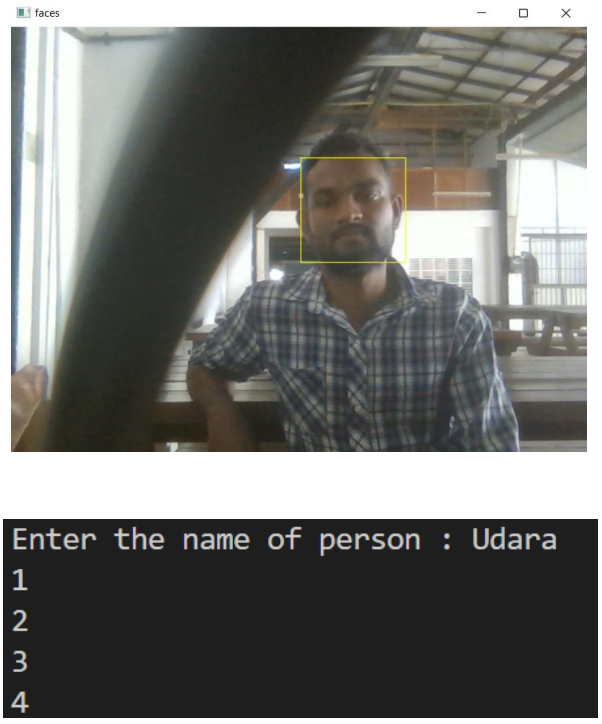

Figure 15: Face registration results

3. Face Recognition

Implemented model accurately and effectively recognizes faces. When the model recognizes a registered person, it displays the corresponding name on the application. If not it displays the name as 'Unknown'.

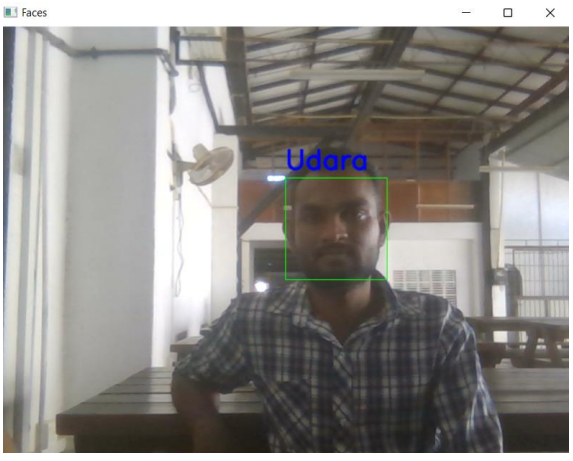

Figure 16: Face recognition results

4.3 Smart Home Application

The forecasting simulation was done for the LSTM algorithm on the following specifictions:16GB RAM, a 4.8 GHz core i7 processor used, and the IDE environment visual studio code and the python language were used.

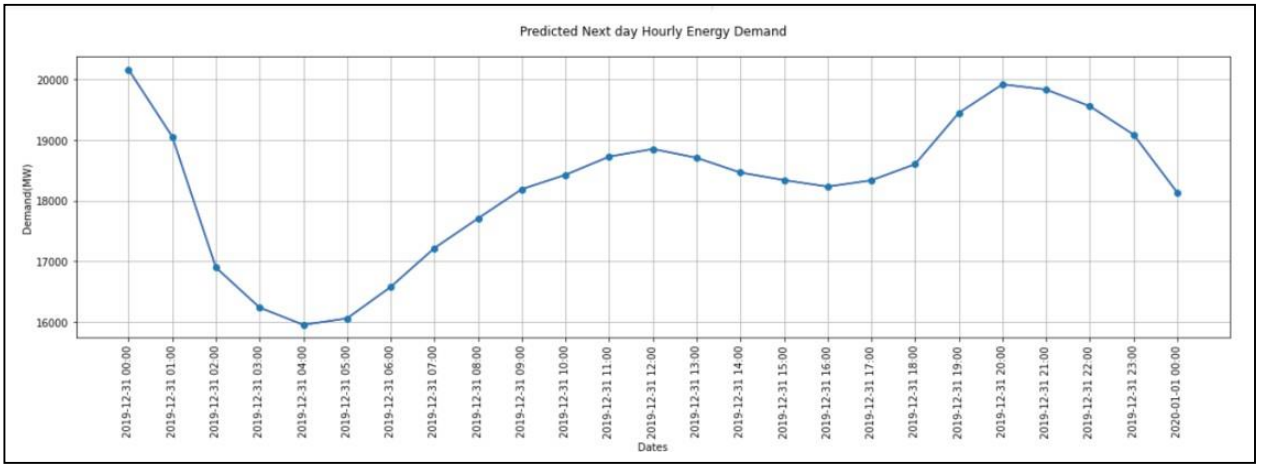

Figure 17: One Day Load Forecasting

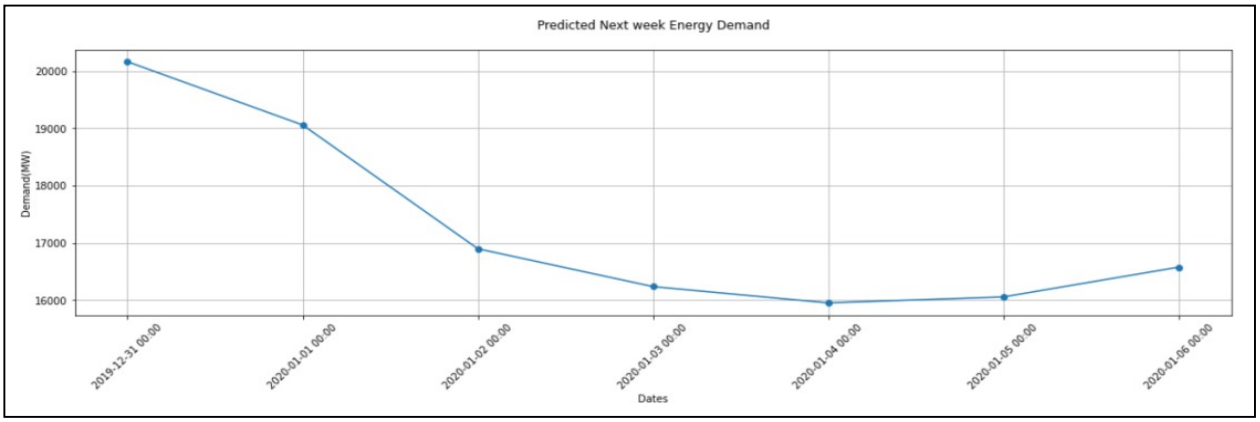

Figure 18: One Week Load Forecasting

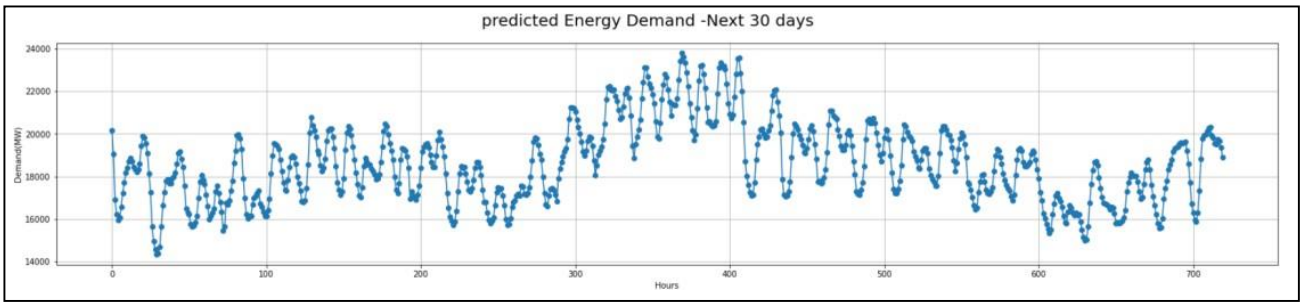

Figure 19: One Month load forecasting

Chapter 5

Future Work

5.1 Smart Home Application

5.1.1 Further improvements in the forecasting model

- Instead of using the pure LSTM model, some other mix models like CNN-LSTM, or Bi-LSTM models can be added with an ensemble learner and check the accuracy score values to verify they are working better than the basic LSTM model. Similarly, there are more modified models with higher accuracy for RNN and CNN. Those also should be tested with the online available dataset for both long-term and short-term predictions.

5.1.2 Further improvements in the Energy monitoring and device controlling prototypes

- Instead of using the pure LSTM model, some other mix models like CNN-LSTM, or Bi-LSTM models can be added with an ensemble learner and check the accuracy score values to verify they are working better than the basic LSTM model. Similarly, there are more modified models with higher accuracy for RNN and CNN. Those also should be tested with the online available dataset for both long-term and short-term predictions.

5.1.3 Further improvements in the Web dashboard

- Improve the more security features on the Web dashboard. Because usually, every SHA and SVA systems has a dashboard to view and control. Therefore, our system as mission critical system, we need to have more advanced security algorithms and techniques to secure the system. For that, a security layer also should lie on top of the application layer. Following are some key components that are needed to design and implement in the future.

  - Authentication

Hence, we measured electricity usage through Energy monitoring device, no one could not be able to alter the device and data (device authentication). Next, the user

who wishes to interact with the system must ensure that he or she is the actual user (user authentication).

- Authorization

  After the user identity is authenticated, the goal of authorization is to assign specific access rights to each user. For example, there might be different types of users in our system. They are mainly two users in the system. They are Super Admins and regular users. Super Users might be the persons who have all access rights and administrators of the systems. Ordinary and regular users might be other people who lives in the home environment with a smaller number of rights. So, both parties have different access rights to differentiate their activities.

- Intrusion Detection

  Intrusion detection systems (IDS)ensures data security by identify attacks prior to it happen. Hence above developed system can be implemented on multiple homes, a collaborative IDS is needed that can be work in distributed environment.

- Privacy

  Using our energy monitor system, we collected energy details and state information. Therefore, middle party could not be able to obtain system users personal details and power consumption details. Hence the privacy of the user must secure using cryptographic technologies.

5.2 Surveillance Video Application

Future Improvements

- Expanding the dataset by collecting a more extensive range of images representing each threatening pose can improve the model's performance and generalization capabilities.
- Exploring deep learning approaches, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), may offer more advanced feature extraction and classification capabilities, potentially enhancing the accuracy of pose detection.
- The ethical implications of deploying surveillance systems should be carefully considered. Privacy concerns and adherence to legal regulations must be prioritized to ensure the responsible.
- The development of a surveillance video application with threatful pose detection holds immense potential for enhancing public safety and security. By leveraging web scraping, MediaPipe for keypoint extraction, and a KNN classifier for pose classification, the application demonstrates accurate detection and classification of threatening poses such as climbing, running, pushing, pulling, and gun shooting. However, further improvements, including increasing the training dataset, exploring advanced deep learning techniques, and implementing real-time capabilities, are recommended to enhance the system's accuracy and usability.
- The developed application represents a significant step toward intelligent surveillance systems, capable of identifying potential threats proactively. Responsible deployment, ethical considerations, and continuous improvement are essential to ensure the application's effectiveness while respecting individual privacy and legal frameworks.

Chapter 6

Conclusions

WLAN edge computing has emerged as a promising approach to enhance the performance of surveillance video processing and smart home applications. By bringing data processing and analysis closer to the source, WLAN edge computing offers reduced latency, improved system efficiency, enhanced privacy and security, and optimized network bandwidth. Leveraging the computational capabilities of edge devices, this approach enables real-time or near-real-time decision-making, efficient resource utilization, and intelligent automation in smart homes. Furthermore, WLAN edge computing facilitates edge intelligence by enabling advanced analytics and machine learning algorithms at the edge, empowering surveillance systems and smart home environments with contextual insights. As research and development in this field continue to advance, WLAN edge computing is expected to play a crucial role in optimizing the performance and responsiveness of surveillance systems and smart homes, paving the way for more efficient and intelligent applications in the future..

Bibliography

[1] Y. Wang, N. Zhang, and X. Chen, “A short-term residential load forecasting model based on LSTM recurrent neural network considering weather features,” MDPI, 11-May-2021. [Online]. Available: https://www.mdpi.com/1996-1073/14/10/2737. [Accessed: 07-Jan2023]

[2] I. Machorro-Cano, G. Alor-Hernández, M. A. Paredes-Valverde, L. Rodríguez-Mazahua, J. L. Sánchez-Cervantes, and J. O. Olmedo-Aguirre, “Hems-IOT: A big data and machine learning-based Smart Home System for Energy Saving,” MDPI, 02-Mar-2020. [Online]. Available: https://www.mdpi.com/1996-1073/13/5/1097. [Accessed: 05-Jan-2023].

[3] Y. Liu, X. Yang, W. Wen, and M. Xia, "Smarter Grid in the 5G ERA: A Framework Integrating Power Internet of things with a cyber physical system," Frontiers, 31-May2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frcmn.2021.689590/full. [Accessed: 29-Dec2022]

[4] Kim, J.-W. et al. (2023) Human pose estimation using MediaPipe pose and optimization method based on a humanoid model, MDPI. Available at: https://www.mdpi.com/2076-3417/13/4/2700 (Accessed: 25 June 2023)

[5] Mahjoub, S. et al. (2022) Predicting energy consumption using LSTM, multi-layer gru and drop-gru neural networks, MDPI. Available at: https://www.mdpi.com/1424-8220/22/11/4062 (Accessed: 25 June 2023)