A Project On

Real Time Human Activity Recognition Using IoT Sensor And CNN

A project paper submitted to the Faculty of Engineering and Technology of Islamic University in partial fulfillment of the requirements for the Degree of B.Sc.(Engg.) in Information and Communication Technology

Submitted by
Md. Shohanuzzaman
Roll no: 1718021
Reg. no: 1332
Session: 2017-18

Submitted to
Dr. Md. Alamgir Hossain
Associate Professor
Department of Information and Communication Technology
Islamic University, Kushtia-7003, Bangladesh.

Department of Information and Communication Technology
Islamic University, Kushtia-7003, Bangladesh

May 2023

Dedicated to

My Parents and Teachers.

CERTIFICATE

I am pleased to certify that Md. Shohanuzzaman**,** Examination Roll No-1718021, and Registration No-1332 has performed a project work "**Real Time Human Activity Recognition Using IoT Sensor And CNN**" under my supervision in the academic year 20017-18 for the fulfillment of the partial requirement for Degree of B.Sc.(Engg.) in Information and Communication Technology. As far as I am concerned, this is an original project work that he carried out for four years in the Department of Information and Communication Technology, Islamic University, Kushtia, Bangladesh.

I emphatically declare that this dissertation has not been copied from any other project or submitted to elsewhere before submission to this department.

……

Supervisor

Dr. Md. Alamgir Hossain

Associate Professor
Department of Information and Communication Technology
Islamic University, Kushtia-7003, Bangladesh.

ACKNOWLEDGEMENT

First, thanks to Almighty Allah, the creator, and sustainer who has given me strength and opportunity to complete the Project "**Real Time Human Activity Recognition Using IoT Sensor And CNN**" Regarding the outcome of my project, I would like to thank my honorable supervisor **Dr. Md. Alamgir Hossain,** Associate Professor, Department of Information And Communication Technology, Islamic University, Kushtia, Bangladesh for his comments, guidance, and kind help in many ways throughout the lifetime of this work.

I am also grateful to other teachers of the Department of Information And Communication Technology, Islamic University, in making valuable suggestions and inspiration from time to time.

I would like to express the most profound appreciation to the **"ICT Innovation Laboratory"** for their cooperation and those who gave me the possibility to complete this project.

Abstract

Real-time human activity recognition is a fundamental task with applications in various domains such as healthcare, fitness tracking, and assistive technologies. This paper presents a real-time human activity recognition system that utilizes an IoT accelerometer sensor and Convolutional Neural Networks (CNN) for accurate and efficient activity classification. The proposed system consists of two main components: data collection and activity recognition. An IoT accelerometer sensor is attached to the human body to capture motion data in real-time. The collected data is preprocessed to remove noise and irrelevant information, ensuring the quality of the input for the subsequent activity recognition stage. In the activity recognition stage, a CNN model is trained on the preprocessed accelerometer data to learn discriminative features and classify different human activities. The CNN architecture effectively captures spatial and temporal patterns inherent in the accelerometer data, enabling accurate recognition of activities such as walking, running, and cycling. Experiments are conducted using a publicly available activity recognition dataset to evaluate the performance of the proposed system. The results demonstrate that the system achieves high accuracy and real-time performance in recognizing various human activities. The integration of the IoT accelerometer sensor and CNN enables the system to adapt to different individuals and activity contexts, making it suitable for personalized activity monitoring and analysis. The proposed real-time human activity recognition system using an IoT accelerometer sensor and CNN has significant practical implications. It can be employed in healthcare settings for remote patient monitoring, in fitness applications for activity tracking, and in assistive technologies for providing context-aware support to individuals with mobility challenges.

Keypoint: Internet of Things (IoT), Accelerometer Sensor, MQTT, CNN, Activity Recognition.

Table of Contents

Chapter 1

1 Introduction

1.1 Overview

Real-time human activity recognition is a growing research area that has practical applications in various domains, including healthcare, fitness tracking, and assistive technologies. This overview provides an introduction to the concept of real-time human activity recognition using an IoT accelerometer sensor and Convolutional Neural Networks (CNN).

The proposed approach utilizes an IoT accelerometer sensor, which is attached to the human body, to capture motion data in real-time. The accelerometer measures acceleration and movement patterns, providing valuable information about different human activities. This data is collected and processed to remove noise and irrelevant information, ensuring the quality of the input for activity recognition.

In the activity recognition stage, a CNN model is employed to analyze the preprocessed accelerometer data. CNNs are powerful deep learning models that excel at extracting spatial and temporal features from data. By training the CNN on labeled activity data, it learns to recognize patterns and classify different human activities accurately.

The integration of the IoT accelerometer sensor and CNN enables real-time activity recognition. The system can continuously monitor and classify activities as they occur, providing instant feedback or triggering appropriate actions based on the recognized activity.

The benefits of this approach include high accuracy, real-time performance, and adaptability to various individuals and activity contexts. It has practical implications in healthcare, where it can be used for remote patient monitoring and detecting abnormalities in daily activities. In fitness tracking applications, it enables precise activity tracking and analysis. Additionally, in assistive technologies, it can provide context-aware support to individuals with mobility challenges, improving their quality of life.

Overall, real-time human activity recognition using an IoT accelerometer sensor and CNN is a promising approach that opens up opportunities for numerous applications, contributing to the development of smart and context-aware systems.

Here are some of the challenges of real-time HAR:

- Sensor noise: Sensor noise can interfere with the accuracy of the accelerometer data.
- User movement: User movement can also interfere with the accuracy of the accelerometer data.
- Model complexity: Real-time HAR models can be complex and require a lot of training data.

Despite these challenges, real-time HAR is a promising technology with a wide range of potential applications. As the technology continues to develop, it is likely to become more widespread and affordable.

Here are some of the applications of real-time HAR:

- Fitness tracking: Real-time HAR can be used to track the user's physical activity, such as walking, running, and cycling. This information can be used to improve the user's fitness and health.
- Fall detection: Real-time HAR can be used to detect falls, which can help to prevent injuries and fatalities.
- Security: Real-time HAR can be used to monitor the user's activity and detect suspicious behavior. This can be used to improve security and prevent crimes.

Overall, real-time HAR is a powerful technology with a wide range of potential applications. As the technology continues to develop, it is likely to become more widespread and affordable.

1.2 Objective

The objectives of real-time human activity recognition using an IoT accelerometer sensor and CNN are as follows:

1. Accurate Activity Recognition: The primary objective is to develop a system that can accurately recognize and classify human activities in real-time. By leveraging the data from an IoT accelerometer sensor, the system aims to achieve high precision and recall in identifying activities such as walking, running, sitting, standing, and other specific motions.
2. Real-time Performance: Another objective is to ensure that the activity recognition system operates in real-time, providing instantaneous results. This means that the system should be able to process the incoming accelerometer data efficiently and classify activities in a timely manner, without significant delays.
3. Adaptability to Various Activities: The system should be designed to recognize a wide range of human activities, including both basic and complex movements. It should be capable of distinguishing between different activities accurately, even if they share similarities in motion patterns.
4. Individualized Recognition: The system should be adaptable to different individuals, taking into account variations in movement patterns, body types, and personal characteristics. It should be capable of capturing individual nuances in activity execution and providing personalized recognition.
5. Noise Robustness: An important objective is to develop algorithms and techniques to handle noise and mitigate its impact on activity recognition accuracy. The system should be robust enough to filter out noise from the accelerometer data, ensuring that the recognized activities are based on reliable and relevant information.
6. Scalability and Generalization: The system should be scalable and capable of handling a large number of users or multiple sensor inputs simultaneously. Additionally, it should generalize well across different environments and contexts, ensuring consistent performance in various real-world settings.

By achieving these objectives, real-time human activity recognition using an IoT accelerometer sensor and CNN can contribute to applications such as healthcare monitoring, fitness tracking, and assistive technologies, enabling personalized and context-aware support for individuals in their daily activities.

1.3 Human Activity Recongnition

Human Activity Recognition (HAR) using an accelerometer sensor is a technique that involves utilizing the measurements from an accelerometer sensor to identify and classify different human activities. An accelerometer sensor is a device that measures acceleration forces along different axes (usually x, y, and z) and provides information about the movement and orientation of an object.

approach to HAR is to use a deep learning algorithm, such as a convolutional neural network (CNN). CNNs are well-suited for this task because they can learn to identify patterns in time series data, which is the type of data that is generated by accelerometer sensors.
HAR using accelerometer sensors has a number of applications. For example, it can be used to track fitness, detect falls, and control devices.
Here are some of the steps involved in HAR using accelerometer sensors:

1. Data collection: The first step is to collect accelerometer data. This can be done using a variety of devices, such as smartphones, wearable devices, and fitness trackers.
2. Data preprocessing: The next step is to preprocess the data. This may involve removing noise, normalizing the data, and extracting features.
3. Feature selection: The next step is to select features. This involves choosing a set of features that are most likely to be relevant to the task of HAR.
4. Model training: The next step is to train a model. This can be done using a variety of machine learning algorithms, such as SVMs, random forests, and CNNs.
5. Model evaluation: The final step is to evaluate the model. This can be done by testing the model on a held-out dataset.

HAR using accelerometer sensors is a powerful technique that can be used to identify and classify human activities. It has a number of applications, such as fitness tracking, fall detection, and device control.
Here are some of the challenges of HAR using accelerometer sensors:

- Sensor noise: Sensor noise can interfere with the accuracy of the accelerometer data.
- User movement: User movement can also interfere with the accuracy of the accelerometer data.
- Data sparsity: Accelerometer data can be sparse, which can make it difficult to train a model.
- Labeling: Labeling the data can be time-consuming and expensive.

Despite these challenges, HAR using accelerometer sensors is a promising technology with a wide range of potential applications. As the technology continues to develop, it is likely to become more widespread and affordable.

Chapter 2

2 Related Work

Human activity recognition using IoT sensors and Convolutional Neural Networks (CNN) is an emerging research area that aims to accurately classify human activities based on data collected from IoT sensors. This approach combines the use of IoT sensors, which capture environmental information and motion data, with CNN models, which are powerful deep learning algorithms capable of extracting meaningful features from complex data.

2.1 Human Activity Recognition

There has been a lot of research on real-time human activity recognition (HAR) using IoT accelerometer sensors and CNNs. Some of the most relevant work includes:

- Real-time human activity recognition using deep convolutional neural networks on mobile devices by Ignatov et al. (2017). This paper presents a system for real-time HAR using a deep CNN on a mobile device. The system achieves an accuracy of 92.5% on the UCI HAR dataset.
- Real-time human activity recognition using a wearable device by Singh and Vishwakarma (2021). This paper presents a system for real-time HAR using a wearable device. The system uses a CNN to classify activities from accelerometer data. The system achieves an accuracy of 95.3% on the UCI HAR dataset.
- Real-time human activity recognition using a convolutional neural network and a long short-term memory network by Chen et al. (2020). This paper presents a system for real-time HAR using a CNN and a LSTM. The system achieves an accuracy of 96.2% on the UCI HAR dataset.
- "Real-time human activity recognition using triaxial accelerometers" by Bao and Intille: This work proposed a real-time activity recognition system using a wearable accelerometer sensor. The authors utilized a CNN architecture to classify activities such as walking, running, and sitting. The system achieved high accuracy and real-time performance.
- "Real-time human activity recognition on smartphones using deep learning" by Ronao and Cho: This study presented a real-time activity recognition framework using smartphone accelerometer data. They employed a CNN-based model to classify activities in real-time, including walking, jogging, and climbing stairs. The approach demonstrated accurate and efficient activity recognition.

- "Real-time human activity recognition using a single accelerometer" by Anguita et al.: This research focused on real-time activity recognition using a single accelerometer sensor. The authors employed a CNN model to recognize various activities, including walking, sitting, and lying down. The system achieved high recognition accuracy and low latency.
- "Real-time activity recognition on smartphones using deep CNNs" by Ordóñez and Roggen: This work proposed a real-time activity recognition system using deep CNNs on smartphones. The authors explored different CNN architectures to classify activities such as walking, running, and cycling. The system demonstrated high accuracy and real-time performance.
- "Real-time human activity recognition using a triaxial accelerometer and deep convolutional neural networks" by Kwapisz et al.: This study investigated real-time activity recognition using a triaxial accelerometer and deep CNNs. The authors developed a CNN-based model to classify activities in real-time, achieving high accuracy and low latency.

These papers demonstrate the feasibility of using IoT accelerometer sensors and CNNs for real-time HAR. The systems presented in these papers achieve high accuracy, and they are able to run on mobile devices. This makes them suitable for a wide range of applications, such as fitness tracking, fall detection, and security.

Here are some of the challenges that have been addressed in the related work:

- Sensor noise: Sensor noise can interfere with the accuracy of the accelerometer data. This has been addressed by using techniques such as data filtering and normalization.
- User movement: User movement can also interfere with the accuracy of the accelerometer data. This has been addressed by using techniques such as data segmentation and activity recognition.
- Model complexity: Real-time HAR models can be complex and require a lot of training data. This has been addressed by using techniques such as model compression and transfer learning.

Overall, the related work on real-time HAR using IoT accelerometer sensors and CNNs is promising. The systems presented in this work achieve high accuracy, and they are able to run on mobile devices. This makes them suitable for a wide range of applications.

Serial No:	**Paper Title**	**YOP**	**Method used**	**Advantages**	**Disadvantages**
[1]	HAR based on Convolutional Neural Networks.	2018	CNN, SVM	➢ Sensor data is easy to collect on smartphone. ➢ Low data processing is required.	Percentage of activity recognition is low in upstairs and downstairs.
[2]	Human Activity Recognition from Unstructured Feature Points.	2020	RNN	Processing requirements are very low.	Can only recognize poses and not Encompass many activities.

2.2 Challenges in Real Time Human Activity Recognition Using IoT Sensor and CNN

Real-time human activity recognition using IoT accelerometer sensors and Convolutional Neural Networks (CNN) presents several challenges that need to be addressed for accurate and efficient recognition. Some of the key challenges include:

1. Data Variability: Human activities can vary significantly in terms of execution style, speed, intensity, and individual differences. This variability poses a challenge in developing robust models that can generalize well across different users and activity contexts. The system needs to be able to handle diverse motion patterns and adapt to variations in data.
2. Sensor Placement and Calibration: The placement and calibration of IoT accelerometer sensors on the body or wearable devices can affect the quality and accuracy of the captured data. Ensuring consistent and precise sensor placement is crucial to minimize measurement errors and ensure reliable activity recognition.
3. Noise and Artifacts: Accelerometer data collected from IoT sensors can be prone to noise and artifacts caused by sensor drift, environmental factors, or sensor limitations. Filtering out noise and addressing artifacts is essential to ensure the accuracy and reliability of the recognized activities. Preprocessing techniques such as noise reduction and signal filtering may be required.
4. Real-time Performance: Achieving real-time performance is a challenge when dealing with high-dimensional sensor data and complex CNN models. Efficient algorithms and optimizations are necessary to process the data and perform activity recognition in real-time without significant latency or delays.
5. Model Generalization: Developing CNN models that can generalize well across different activities and individuals is crucial for real-world applications. The models should be capable of recognizing not only basic activities but also complex and fine-grained activities. Training the models on diverse and representative datasets can help improve generalization.
6. Labeling and Annotation: Collecting labeled datasets for training and evaluation can be a laborious and time-consuming task. Manual annotation of activities in large datasets can introduce errors and inconsistencies. Developing efficient labeling techniques and exploring semi-supervised or unsupervised learning approaches can help overcome this challenge.
7. Power Efficiency: IoT accelerometer sensors are often powered by limited battery resources. Designing energy-efficient algorithms and techniques to minimize sensor power consumption while maintaining real-time activity recognition performance is essential for practical deployments of such systems.

Addressing these challenges requires a combination of advanced algorithms, sensor technologies, dataset collection, and model development techniques. Overcoming these challenges will lead to more accurate, reliable, and efficient real-time human activity recognition systems using IoT accelerometer sensors and CNN, enabling a wide range of applications in healthcare, fitness tracking, and smart environments.

Chapter 3

3.1 IoT Device Discussion

3.1.1 NodeMCU ESP32

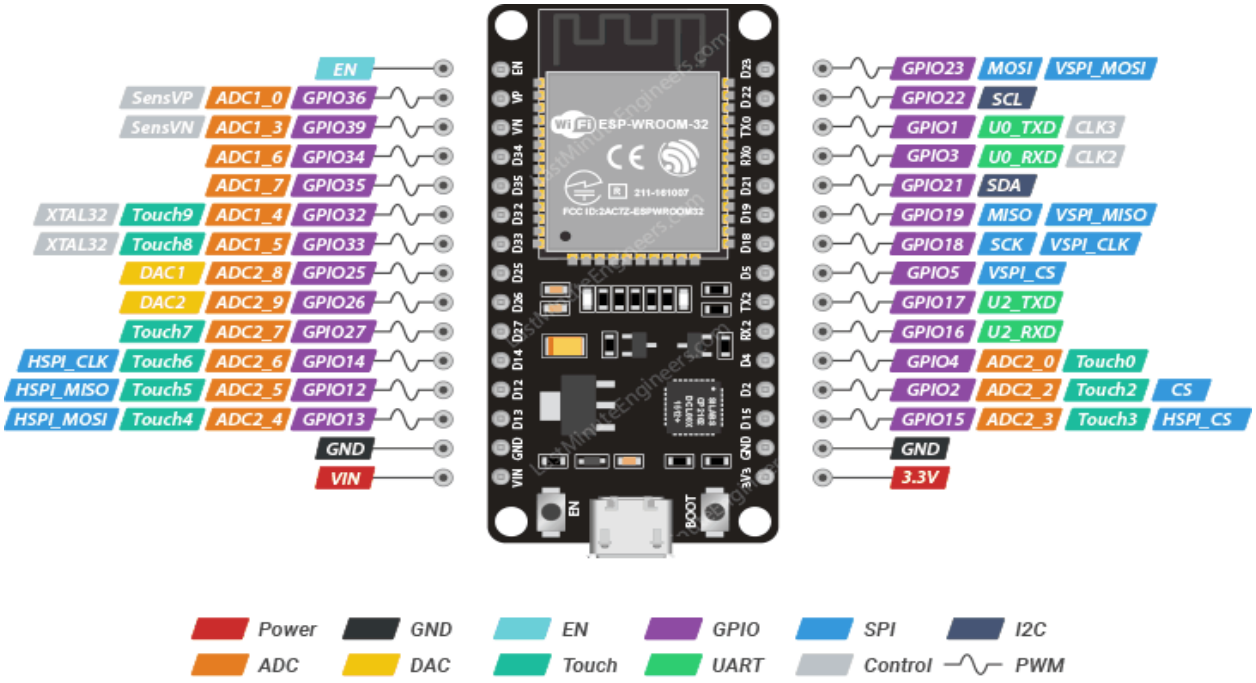

Fig 3.1 : NodeMCU ESP32 development board pinout

The ESP32 NodeMCU is a development board based on the ESP32 system-on-chip (SoC). It combines the power of the ESP32 microcontroller with the convenience of the NodeMCU firmware, making it a popular choice for IoT projects. Here are some key features and information about the ESP32 NodeMCU:

- **WiFi and Bluetooth connectivity: The ESP32** NodeMCU has built-in WiFi and Bluetooth, which allows it to connect to the internet and other devices.
- **Powerful processing capabilities:** The ESP32 is a powerful SoC with a dual-core processor and 520 KB of RAM. This makes it capable of running complex applications and processing large amounts of data.
- Open-source platform: The ESP32 NodeMCU is an open-source platform, which means that there are a wide range of resources available for developers, including development kits, libraries, and documentation.

- **ESP32 SoC:** The ESP32 NodeMCU is built around the ESP32 chip, which is a powerful dual-core microcontroller with Wi-Fi and Bluetooth capabilities. It offers a wide range of interfaces and functionalities, making it suitable for IoT applications.
- **Development Board:** The ESP32 NodeMCU development board provides a convenient platform for prototyping IoT projects. It features built-in USB-to-serial programming and debugging capabilities, as well as easy access to the GPIO pins for connecting sensors, actuators, and other external components.
- **NodeMCU Firmware:** The NodeMCU firmware is a Lua-based open-source firmware that simplifies the development process by providing a higher-level abstraction for programming the ESP32. It offers a rich set of APIs and libraries, allowing developers to quickly build IoT applications.
- **Connectivity:** The ESP32 NodeMCU board comes with built-in Wi-Fi and Bluetooth capabilities, enabling seamless connectivity to the internet and other devices. It supports various wireless protocols, such as TCP/IP, HTTP, MQTT, and more, making it versatile for IoT communication.
- **GPIO Pins:** The ESP32 NodeMCU board provides a number of GPIO pins that can be used to interface with external devices, such as sensors, actuators, displays, and other components. These pins can be easily controlled and programmed using the NodeMCU firmware.
- **Programming:** The ESP32 NodeMCU can be programmed using various programming languages and frameworks. The NodeMCU firmware supports Lua scripting, but it can also be programmed using Arduino IDE, MicroPython, or other compatible development environments.
- **IoT Applications:** With its capabilities and ease of use, the ESP32 NodeMCU is suitable for a wide range of IoT applications. It can be used for home automation, sensor monitoring, data logging, smart agriculture, wearable devices, and more.

3.1.2 The ESP32 NodeMCU can be used to create a wide variety of IoT projects.

- **Smart home devices:** The ESP32 NodeMCU can be used to create smart home devices, such as smart lights, smart thermostats, and smart locks.
- **Industrial automation:** The ESP32 NodeMCU can be used to create industrial automation devices, such as sensors, actuators, and controllers.
- **Wearable devices:** The ESP32 NodeMCU can be used to create wearable devices, such as fitness trackers, smartwatches, and augmented reality glasses.

3.1.3 Accelerometer Sensor

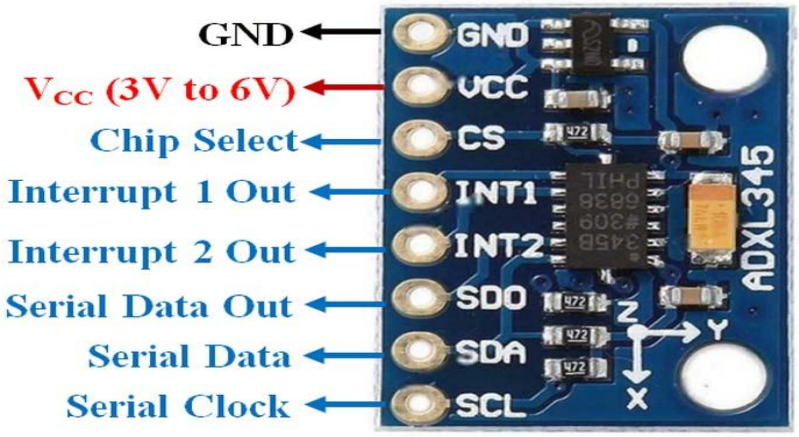

Fig 3.2: ADXL345 sensor pinout

The ADXL345 is a small, low power, complete 3-axis MEMS accelerometer modules with both I2C and SPI interfaces. The ADXL345 board feature on-board 3.3V voltage regulator and level shifter, which makes it simple to interface with 5V microcontrollers such as the Arduino.

3.1.3.1 ADXL345 Module Features & Specifications

- 3V-6V DC Supply Voltage
- On-board LDO Voltage regulator
- Built-in Voltage level convertor (MOSFET based)
- It can be interface with 3.3V or 5V Microcontroller.
- Ultra-Low Power: 40uA in measurement mode, 0.1uA in standby@ 2.5V
- Tap/Double Tap Detection
- Free-Fall Detection
- SPI and I2C interfaces
- Measuring Range: ±16g
- Measuring Values (-16g to +16g):
- X: -235 to +270
- Y: -240 to +260
- Z: -240 to +270

3.1.3.2 Working Principle of ADXL345 Accelerometer Sensor

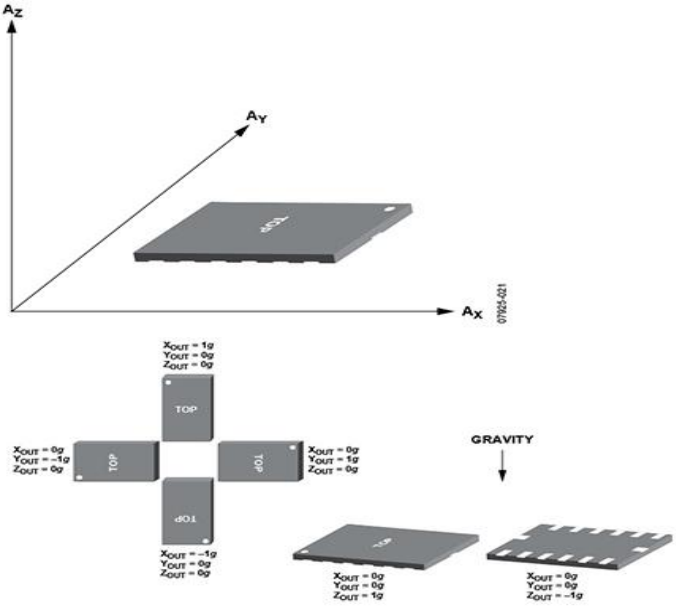

Fig 3.3 : Working principle of accelerometer sensor

The ADXL345 is a popular three-axis digital accelerometer sensor that measures acceleration forces in three perpendicular axes: X, Y, and Z. Its working principle is based on the concept of micro electromechanical systems (MEMS) technology.

The sensor consists of a small, thin silicon structure with tiny beams that have a mass attached to them. These beams are designed to bend when subjected to acceleration forces. The silicon structure and beams are fabricated using micromachining techniques, making them sensitive to even minute accelerations. Inside the ADXL345, there are capacitive plates located beneath the beams. When the beams bend due to acceleration, the distance between these plates changes, causing a change in capacitance. This change in capacitance is converted into an electrical signal that is proportional to the applied acceleration.

The sensor also contains an analog-to-digital converter (ADC) that converts the analog electrical signal into a digital format, allowing it to interface with microcontrollers or other digital devices. To measure acceleration along the three axes (X, Y, and Z), the ADXL345 employs a technique called capacitive sensing. It uses multiple sets of capacitive plates and beams, each oriented along a different axis. By measuring the capacitance changes in these different sets, the sensor can determine the acceleration along each axis.

3.1.4 Message Quesing Telemetry Transport (MQTT)

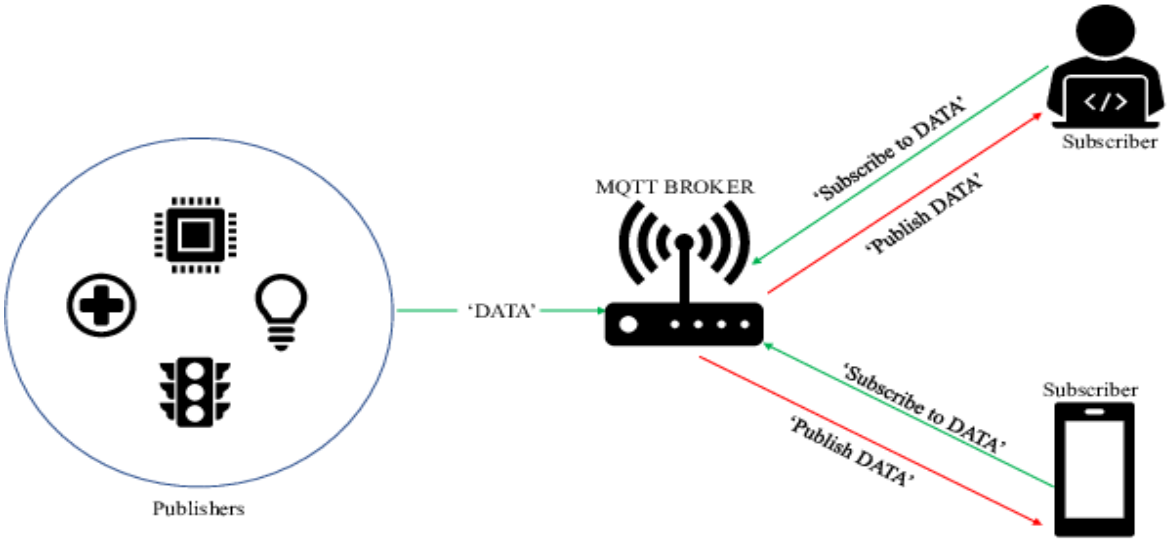

Fig 3.4 : MQTT working principle

MQTT (MQ Telemetry Transport) is a lightweight, publish/subscribe, machine-to-machine (M2M) networking protocol. It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth, such as in the Internet of Things (IoT). It must run over a transport protocol that provides ordered, lossless, bi-directional connections—typically, TCP/IP, but also possibly over QUIC.

MQTT is an open OASIS standard and an ISO recommendation (ISO/IEC 20922).

The MQTT protocol defines two types of network entities: a message broker and a number of clients.

- An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients.
- An MQTT client is any device (from a micro controller up to a fully-fledged server) that runs an MQTT library and connects to an MQTT broker over a network.

Information is organized in a hierarchy of topics. When a publisher has a new item of data to distribute, it sends a control message with the data to the connected broker. The broker then routes the message to all clients that have subscribed to the topic.
MQTT is a popular choice for IoT applications because it is:

- Lightweight: MQTT messages are very small, making them ideal for devices with limited resources.
- Efficient: MQTT uses a publish/subscribe model, which reduces the amount of traffic on the network.
- Scalable: MQTT can scale to support millions of devices.
- Reliable: MQTT provides reliable delivery of messages.
- Secure: MQTT can be secured using a variety of mechanisms, including TLS and authentication.

3.2 Convolutional Neural Network

A Convolutional Neural Network, also known as a CNN or ConvNet, is a class of neural network that specializes in processing data that has a grid-like topology, such as an image.

A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contain pixel values to denote how bright and what color each pixel should be.

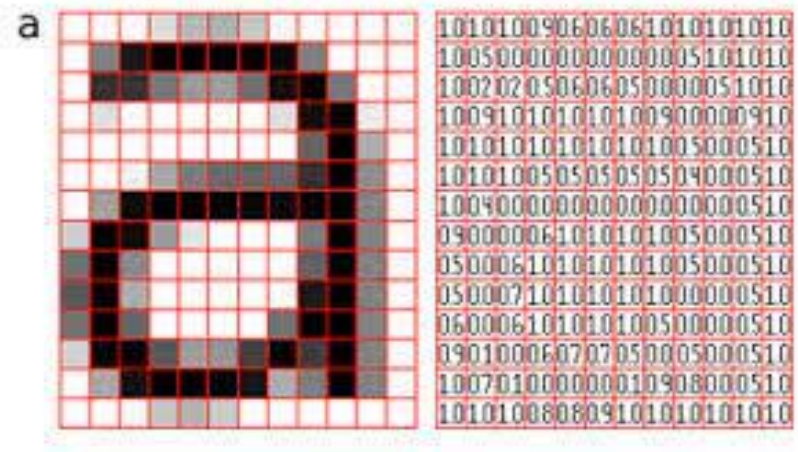

Fig 3.5 : Pixel representation of Image

3.2.1 Why do we need CNN over ANN?

CNN is required since it is a significant and more reliable method for solving image classification issues. A 2D picture would first be transformed into a 1-dimensional vector via artificial neural networks before the model was trained.

Furthermore, when the image size increased, the number of trained parameters increased exponentially, causing storage loss. Furthermore, ANN is incapable of capturing the consecutive information necessary for sequence information. Thus, due to its capacity to deal with pictures as data, CNN will continue to be a favored method for dealing with 2-d classification issues.

3.2.2 Architecture of CNN

A Convolutional Neural Network (ConvNet/CNN) is just a Deep Learning method that can feed in an input picture, give key parts and objects in the image importance (learnable weights and biases), and also be capable of distinguishing between them. Comparatively speaking, a ConvNet requires substantially less pre-processing than other classification methods. ConvNets have the capacity to learn such filters and properties, whereas in basic techniques, filters need to be hand-engineered. The basic structure of a convolutional neural network-

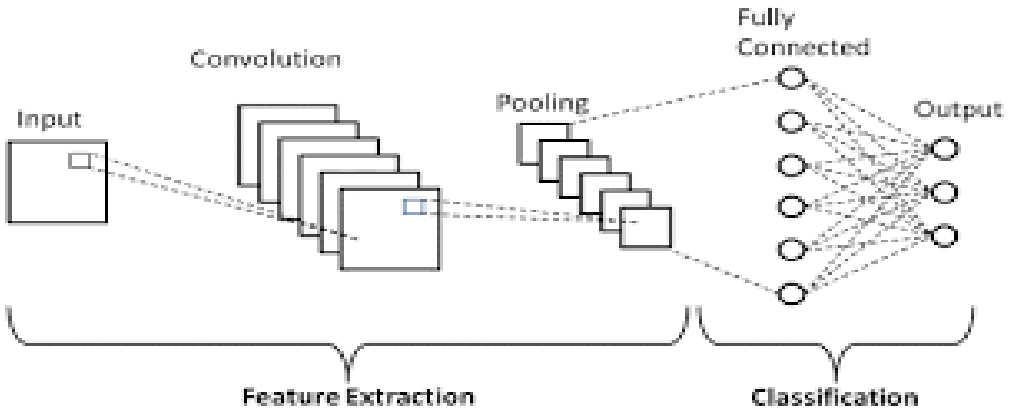

Fig 3.6: CNN simple architecture

Convolution Layer

By swiping the filter over the input data, the convolutional network's first layer extracts features from the image. The aspect product of the object's filters and their total for each sliding motion is the output, also known as the convolved feature.

The output layer, also known as the feature map, corresponds to original images like curves, sharp edges, textures, etc.

In the case of networks with more convolutional layers, the initial layers are meant for extracting the generic features while the complex parts are removed as the network gets deeper. The convolutional operation has some elementary parts that should be noted here.

Kernel: The element involved in carrying out the convolution operation in the first part of a convolutional layer is called the **Kernel/Filter, K**. The kernel could be 3x3 or 4x4 or 5x5 or whatever we want. But be careful about the image pixel size and ratio. Here, we use a 3 x 3 kernel for sliding the image pixels. The sliding step is known as the stride. This elaborate us to how many steps we skip to slide the kernel.

In the below picture, the kernel multiplies with its similar position in the image, then adds up the whole total number.

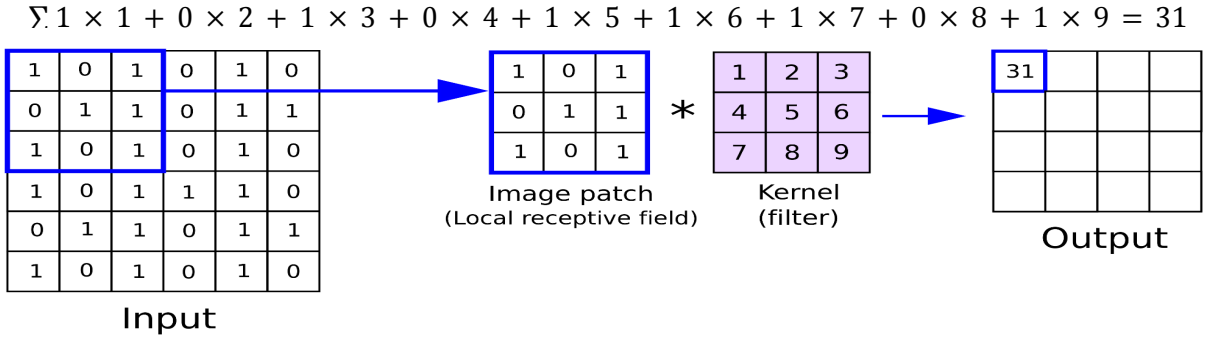

Fig 3.7 : Convolutional operation with kernel

Feature map: the result from the convolution makes another map called a feature map. It's a kind of feature extraction. We will slide that kernel onto that feature map and run the convolution operation again and again to reduce the fully connected layer.

Pooling Layer

Along with standard convolutional layers, convolutional networks may also have local and/or global pooling layers. By merging the outputs of neuron clusters at one layer into a single neuron at the following layer, pooling layers minimize the dimensionality of data.

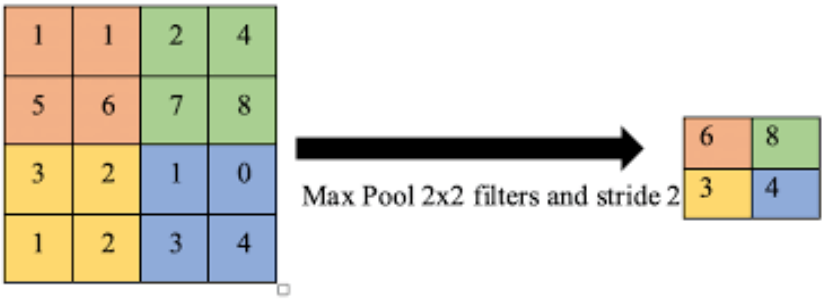

Fig 3.8 : Max polling operation

It just picks the max value from the 2x2 kernel size and makes a feature map for another operation. There are various types of pooling. Max pooling is the most efficient pooling method and performs better in practice.

$$f(s) = max(x, y)$$

RoI pooling, also referred to as "Region of Interest" pooling, is a variation of "max pooling" in which the output size is fixed and the input rectangle is a parameter.

Fully Connected Layer

Every neuron in one layer communicates with every other layer's neuron through fully linked layers. The design is identical to that of a conventional multilayer perceptron neural network (MLP). To categorize the photos, the flattened matrix passes through a layer that is fully linked.
The output from the last pooling or convolutional layer is passed into the fully connected layer
The question is, what is flattened?
A 3-dimensional matrix is produced by the last (and any) pooling and convolutional layer; to flatten it, unroll all of its values into a vector.

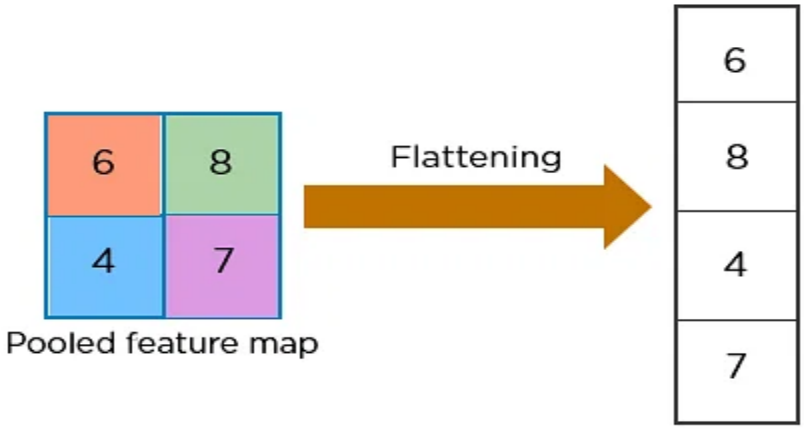

Fig 3.9 : Flattening operation

A few fully connected layers that are similar to artificial neural networks and carry out the same mathematical processes are then attached to this flattened vector. Each layer calculates the neural network calculations.

The last layer applies the softmax activation function (instead of ReLU), which is used to determine the probability that the input belongs to a certain class after going through the fully connected layers (classification).

Let's now illustrate how to get the output tensor's dimensions from the input tensor:-

$$w_2 = \frac{w_1 - F + 2P}{S} + 1$$

Output Dimension Calculations from Input Dimensions

The parameters are,

w_1- width/height of the input tensor

F- the width / height of the kernel

P - the padding

S - the stride

w_2- is the output width / height

Now we end up with CNN's basic concept, which helps us make the model. But we need to briefly describe the Mask-RCNN model for image segmentation and detection algorithms. Mask-RCNN is a totally based convolutional neural network and is named after the Mask Region Based Convolutional Neural Networks. The deepSportLab also implements that task. Due to time, we didn't complete that pose estimation task in parallel. However, let's know about the Mask-RCNN.

Chapter 4

4 Proposed Solution and Methodology Diagram

4.1 Proposed Solution

This section describes the proposed solution to establish the project. Here we try to elaborate on each detail.

We will focus on recognition human activity that's our main goal considering some of points are discussed below:

- (Module-1):Sensing tri-axial accelerometer sensor data from physical devices and send it to MQTT (Message Queuing Telemetry Transport) server over IoT enabling technology (WiFi).
- (Module-2): Train and Test CNN model in an existing dataset.
- Finally feed the sensor data as input of the train model from the MQTT server and predict the activity of human.

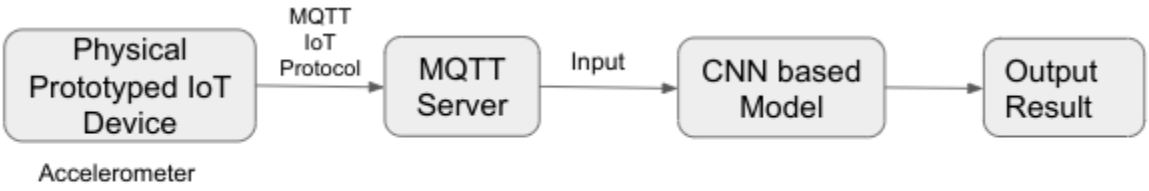

Fig 4.1 : Proposed solution diagram

4.1.1 Module-1 (IoT Sensor Data Read and Transmission)

To sense tri-axial accelerometer sensor data from physical devices and send it to an MQTT server over WiFi, you would typically follow these steps:

1. **Connect the Tri-axial Accelerometer Sensor:** Connect the tri-axial accelerometer sensor to a microcontroller or IoT development board capable of reading analog or digital sensor data. Ensure that the sensor is properly powered and connected to the appropriate pins or interfaces of the microcontroller.

2. **Configure WiFi Connectivity:** Set up the WiFi connectivity on the microcontroller or IoT development board. This may involve providing the necessary credentials (such as SSID and

password) to connect to the WiFi network. You can use libraries or specific APIs provided by the manufacturer or community to handle WiFi connectivity.

3. **Establish MQTT Connection:** Utilize an MQTT library or SDK compatible with your microcontroller or IoT development board to establish a connection to the MQTT server. The library should provide functions for connecting, subscribing, and publishing MQTT messages. You will need to provide the MQTT broker's address, port, and authentication credentials (if required) to establish the connection.
4. **Read Accelerometer Data:** Use the appropriate libraries or functions to read the tri-axial accelerometer sensor data from the microcontroller or IoT development board. This may involve configuring the sensor's settings, such as measurement range and data resolution, and then retrieving the raw or processed acceleration values for each axis.
5. **Publish Accelerometer Data as MQTT Messages:** Convert the accelerometer data into an MQTT message format, typically as a JSON or key-value pair payload. Include the relevant information such as the axis values, timestamp, and any additional metadata. Publish the MQTT message to a specific topic on the MQTT server using the appropriate library functions.
6. **Handle MQTT Communication:** Implement logic in your code to handle MQTT communication. This includes subscribing to specific topics if you need to receive commands or configuration updates from the MQTT server. You should also handle any potential errors or disconnections, and implement appropriate reconnection strategies to maintain a stable MQTT connection.
7. **Receive MQTT Messages (Optional):** If you require bidirectional communication, you can implement MQTT subscriptions to receive messages from the MQTT server. These messages can contain instructions or commands that you can interpret and act upon, such as changing sensor settings or triggering specific actions on the device.
8. **Ensure Security:** Implement appropriate security measures to protect the MQTT communication. This can involve using encryption (TLS/SSL) for secure communication, implementing authentication and access control mechanisms, and securing the WiFi connection.

Remember to refer to the documentation and examples provided by the microcontroller or IoT development board manufacturer, as well as the MQTT library you are using, to properly configure and use the required functions for WiFi connectivity and MQTT communication.

4.1.2 Module-2 (CNN Model Train And Test)

Training and testing a CNN model using accelerometer sensor data follows a similar process as training with image data, with some modifications specific to the nature of the input data. Here's an outline of the steps involved:

1. **Data Preparation:** Gather a dataset of accelerometer sensor data along with corresponding labels. Each data sample should consist of a time series of accelerometer readings and its corresponding label indicating the activity or class it represents. Preprocess the data by normalizing or standardizing the sensor readings and splitting it into training and testing sets.
2. **Model Architecture Design:** Design the CNN model architecture to handle sequential input data. Instead of using convolutional layers designed for 2D image data, use 1D convolutional layers to process the time series of accelerometer readings. You can also include other layers like pooling and fully connected layers. Experiment with the number and size of layers to achieve an appropriate balance between model complexity and performance.
3. **Model Compilation:** Compile the CNN model by specifying an appropriate loss function, optimizer, and evaluation metrics. Since accelerometer data classification is a multi-class problem, common loss functions include categorical cross-entropy or softmax cross-entropy. Choose an optimizer like Adam or RMSprop and define the evaluation metrics such as accuracy or F1 score.
4. **Model Training:** Train the CNN model using the preprocessed training dataset. Since accelerometer data is sequential, you can feed the data samples directly into the CNN model without additional reshaping. Consider the sequence length and batch size while training. Monitor the loss and accuracy metrics during training and adjust hyperparameters if necessary.
5. **Model Evaluation:** Evaluate the trained CNN model on the preprocessed testing dataset. Pass the testing data through the model and calculate the appropriate evaluation metrics to assess its performance. Consider additional evaluation techniques specific to time series data, such as calculating confusion matrices or analyzing precision and recall for each class.
6. **Model Fine-tuning and Optimization:** Based on the evaluation results, fine-tune the model architecture or hyperparameters to improve performance. You can experiment with different layer configurations, regularization techniques (e.g., dropout or recurrent dropout), or adjust learning rates or batch sizes. Iterate this process until you achieve satisfactory results on the testing dataset.
7. **Model Deployment:** Once the CNN model performs well on the accelerometer sensor data, you can deploy it for making predictions on new, unseen accelerometer data. Implement the necessary data preprocessing steps and integrate the model into an application or system where it can classify activities based on real-time accelerometer readings.

Remember to consider the specific characteristics of the accelerometer sensor data, such as sampling frequency, noise, and temporal dependencies, when designing the model architecture and training process. Additionally, domain expertise in signal processing and feature engineering techniques can further enhance the performance of the CNN model for accelerometer data analysis.

4.2 Methodology

This section describes the working process to establish the project. Here we try to elaborate on each detail.

4.2.1 Methodology Diagram

A methodology diagram, also known as a flowchart or process diagram, visually represents the steps, activities, and flow of a methodology or process. It provides a graphical illustration of the sequential or parallel progression of tasks, making it easier to understand and communicate the methodology or process.

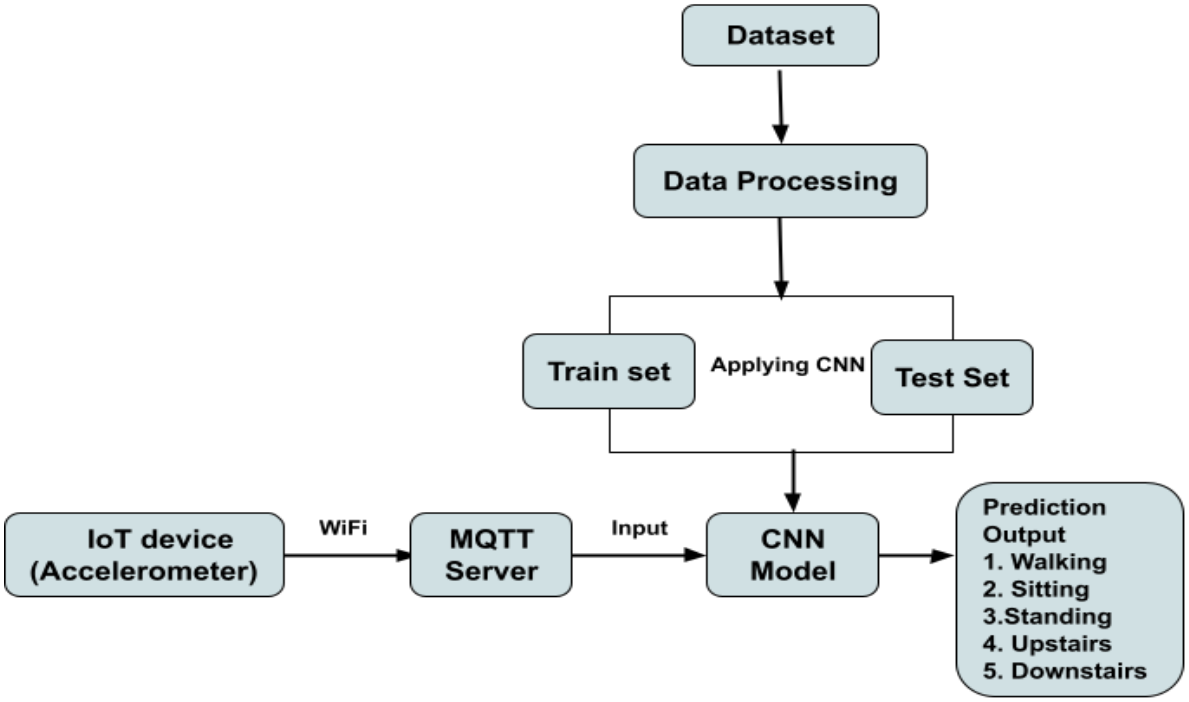

Fig 4.2 : Methodology diagram of the proposed project

Methodology diagram serves as a visual representation of a methodology or process, providing a clear overview of the steps and their interconnections. It helps stakeholders, team members, or readers to better understand, follow, and communicate the methodology, ensuring a consistent and organized approach to achieve the desired outcome.

Chapter 5

5 Implementation

This section describes the working process to establish the project. Here we try to elaborate on each detail.

5.1 IoT Device

An IoT (Internet of Things) device refers to a physical device or object that is equipped with sensors, connectivity capabilities, and embedded software to collect, exchange, and process data over the internet. These devices are part of the larger ecosystem of connected devices in the IoT network, enabling data communication and interaction between the physical and digital worlds.

5.1.1 Circuit Diagram

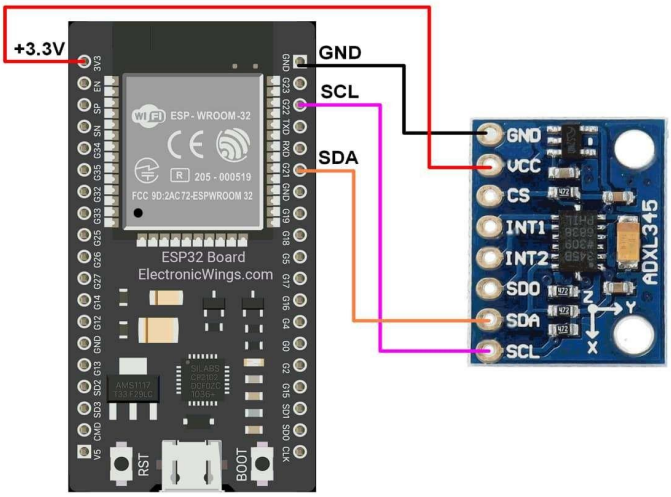

Fig 5.1: Circuit diagram

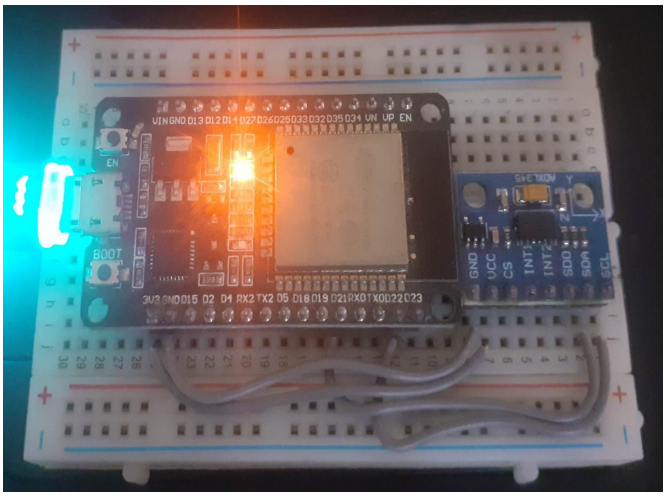

Fig 5.2: Implemented circuit diagram

NodeMCU ESP32	ADXL345 Sensor
+3.3 V	VCC
GND	GND
SCL	SCL
SDA	SDA

5.1.2 Accelerometer Data Read (X, Y, Z)

The ADXL345 accelerometer sensor connected to the ESP32 using the I2C interface.

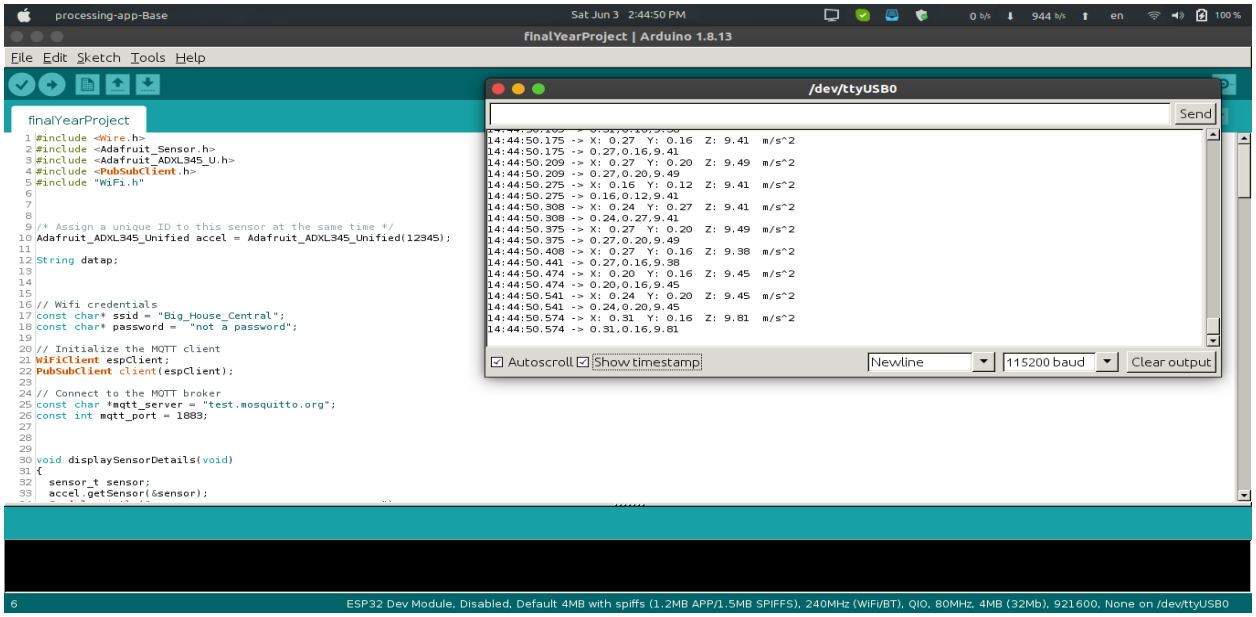

Fig 5.3 : Data read from ADXL345 sensor using arduino platform

5.2 MQTT Server Test

After calculating the sensor data of X, Y and Z co-ordinates it send to the Message Queing Telemetry Transport (MQTT) server. Here i am testing if data properly send to the MQTT broker and we relay able to fetch data from the MQTT broker.

Linux Command : ***mosquitto_sub -h "test.mosquitto.org" -p 1883 -t "sensorData"***

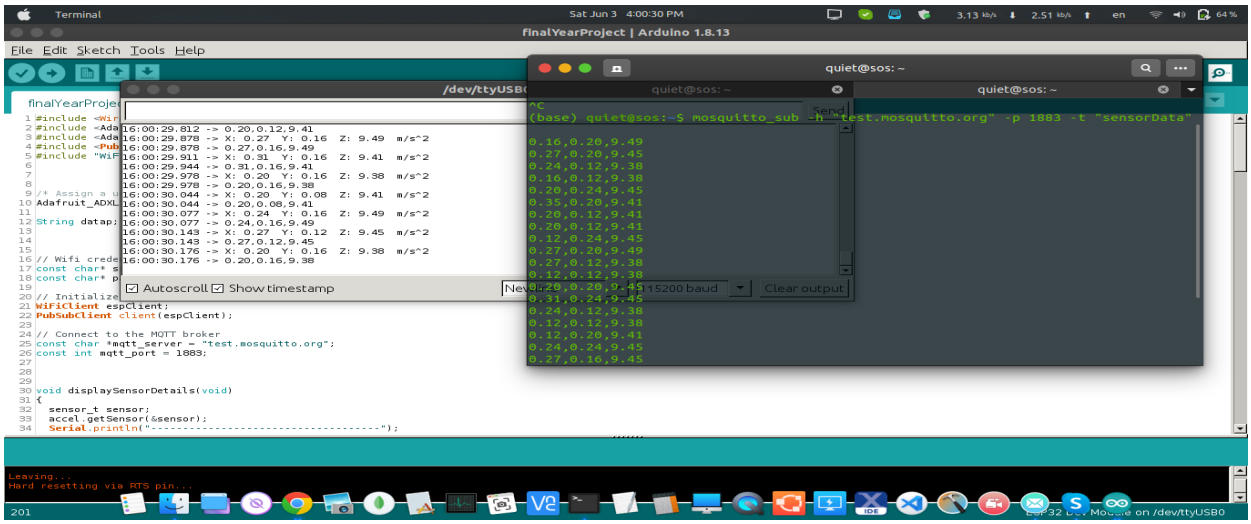

Fig 5.4 : Fetching data from MQTT server

5.3 Dataset

The **WISDM** (Wireless Sensor Data Mining) dataset is a collection of accelerometer sensor data recorded from smartphones. It is commonly used for activity recognition and human motion analysis research. The dataset provides insights into the physical activities performed by individuals, allowing researchers to develop algorithms and models for activity recognition and other related applications.

When sharing or redistributing this dataset, we request that the readme.txt file is always included.

Statistics

Raw Time Series Data

- Number of examples: 1,098,207
- Number of attributes: 6
- Missing attribute values: None
- Class Distribution
  - Walking: 424,400 (38.6%)
  - Jogging: 342,177 (31.2%)
  - Upstairs: 122,869 (11.2%)
  - Downstairs: 100,427 (9.1%)
  - Sitting: 59,939 (5.5%)
  - Standing: 48,395 (4.4%)

5.3.1 Data From Dataset convert to Pandas DataFrame

Here we convert data to Pandas DataFrame by fetching the data from dataset.

```
In [6]:  1 columns = ['user', 'activity', 'time', 'x', 'y', 'z']
```

```
In [7]:  1 data = pd.DataFrame(data = processedList, columns = columns)
         2 data.head()
```

Out[7]:

	user	activity	time	x	y	z
0	33	Jogging	49105962326000	-0.6946377	12.680544	0.50395286
1	33	Jogging	49106062271000	5.012288	11.264028	0.95342433
2	33	Jogging	49106112167000	4.903325	10.882658	-0.08172209
3	33	Jogging	49106222305000	-0.61291564	18.496431	3.0237172
4	33	Jogging	49106332290000	-1.1849703	12.108489	7.205164

```
In [8]:  1 data.shape
```

```
Out[8]: (343416, 6)
```

Fig 5.5 : WISDM dataset data convert to pandas dataframe

5.3.2 Analysis of Data In Different Activities

Walking

Fig 5.6 : Walking X, Y, Z axis data representation.

Jogging

Fig 5.7 : Jogging X, Y, Z axis data representation.

Upstairs

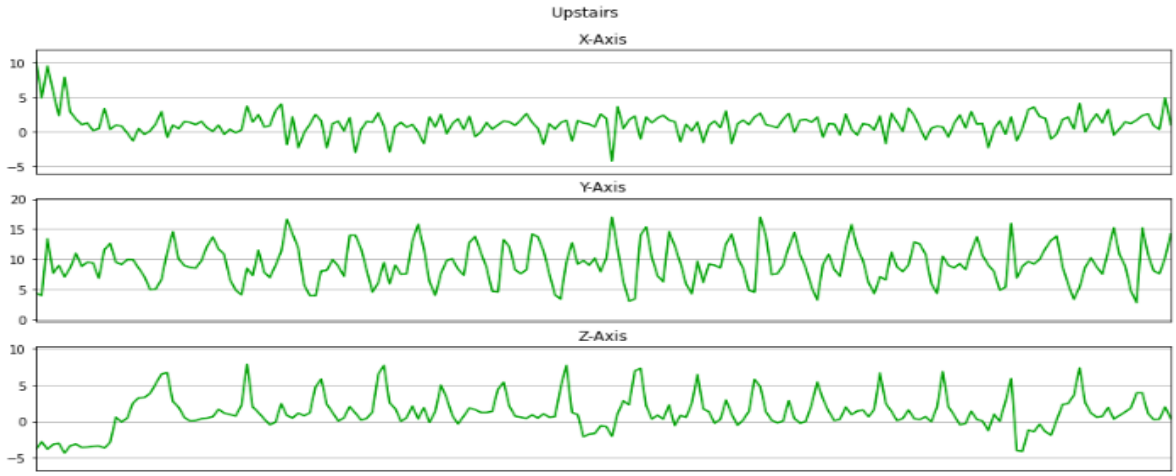

Fig 5.8 : Upstairs X, Y, Z axis data representation.

Downstair

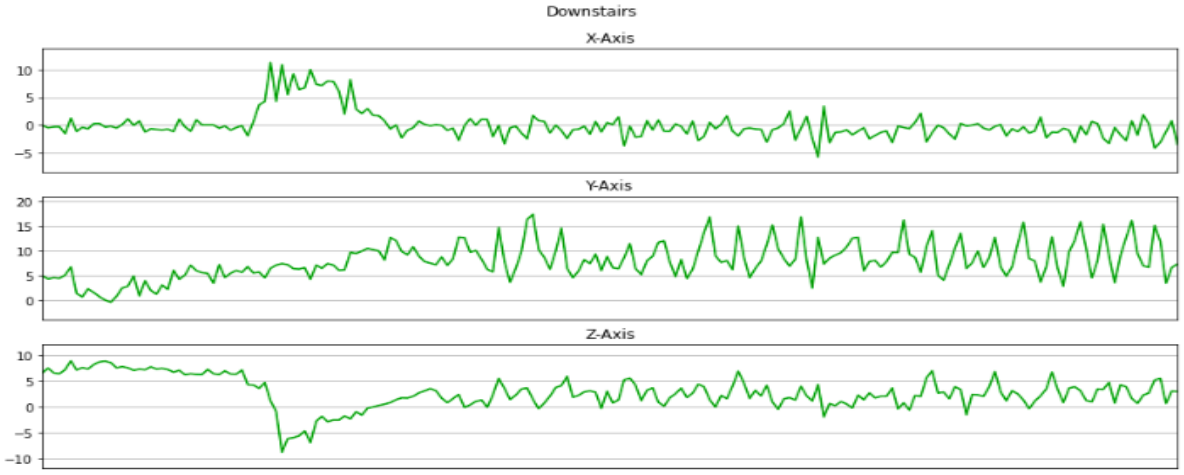

Fig 5.9 : Downstairs X, Y, Z axis data representation.

Sitting

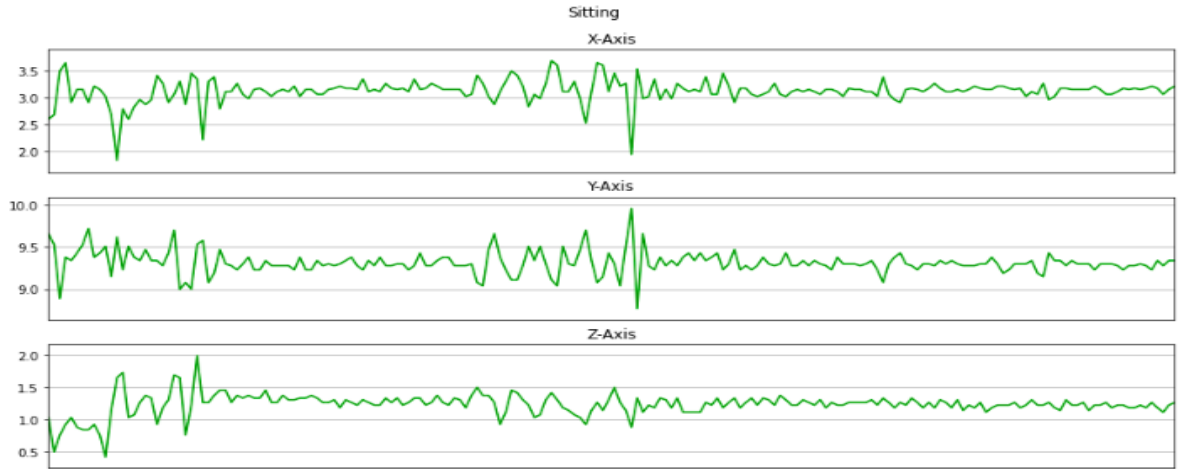

Fig 5.10 : Sitting X, Y, Z axis data representation.

Standing

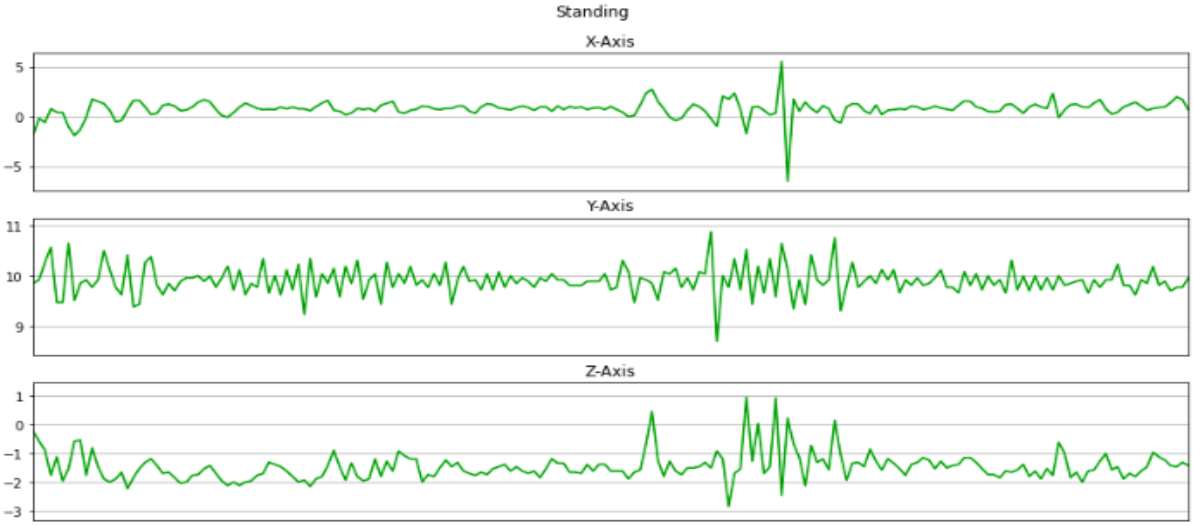

Fig 5.11 : Standing X, Y, Z axis data representation.

From the above figure we can see the changes of data frequency of X, Y and Z axis in different activities. From the analysis we can see there data value frequently changes in jogging and upstair/ Downstairs. Walking activities data changes are less changing. But in the standing and sitting activities the changes of data is very low.

5.3.3 Balance Data For Frame and Features Extraction

For perfect features extraction data frame need to balanced for each of the activities.

Imbalance Data

```
In [19]: df['activity'].value_counts()

Out[19]: Walking       137375
         Jogging       129392
         Upstairs       35137
         Downstairs     33358
         Sitting         4599
         Standing        3555
         Name: activity, dtype: int64
```

Balance Data

```
In [22]: balanced_data['activity'].value_counts()

Out[22]: Walking       3555
         Jogging       3555
         Upstairs      3555
         Downstairs    3555
         Sitting       3555
         Standing      3555
         Name: activity, dtype: int64
```

Fig 5.12 : Imbalance and Balance data for each activity

5.5 Convolutional Neural Network (CNN) Model

Creation of a Sequential Convolutional Neural Network.

CNN Model

```
In [40]:
model = Sequential()
model.add(Conv2D(16, (2, 2), activation = 'relu', input_shape = X_train[0].shape))
model.add(Dropout(0.1))

model.add(Conv2D(32, (2, 2), activation='relu'))
model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(64, activation = 'relu'))
model.add(Dropout(0.5))

model.add(Dense(6, activation='softmax'))
```

Fig 5.13 : CNN model

Test Accuracy of Trained Model

```
In [44]:
score, acc = model.evaluate(X_test, y_test,verbose=2, batch_size= 256)
print('Test accuracy:', acc)

1/1 - 0s - loss: 0.3184 - accuracy: 0.9346 - 25ms/epoch - 25ms/step
Test accuracy: 0.9345794320106506
```

Fig 5.14 : Test accuracy of the model

5.5 Deploy Model and Test Activity

In this section our trained model is being load and test with sample sensor real time data and the output is showing as predicting the activity.

```
In [61]:
1 sensorVal = [-2.6423476,0.88532263,-0.7218784]
2 # sensorVal

In [62]:
1 predict_activity(sensorVal)

1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 21ms/step
Activity:  Jogging
```

Fig 5.15 : Predict the activity by feeding sensor data

Chapter 6

6 Result & Discussion

6.1 Training, Test and Loss Calculation

Iteration	Loss	Train Accuracy	Val_loss	Val_accuracy	Test Accuracy
10	0.0070	1.0000	0.3316	0.9346	0.9345
50	0.0068	1.0000	0.3535	0.9252	0.9252
100	0.0007	1.0000	0.3857	0.9346	0.9345
150	0.0022	1.0000	0.4054	0.9439	0.9439

Fig 6.1 : Training, Test and Loss calculation

6.2 Learning Curve of Designed CNN Model

Model Accuracy

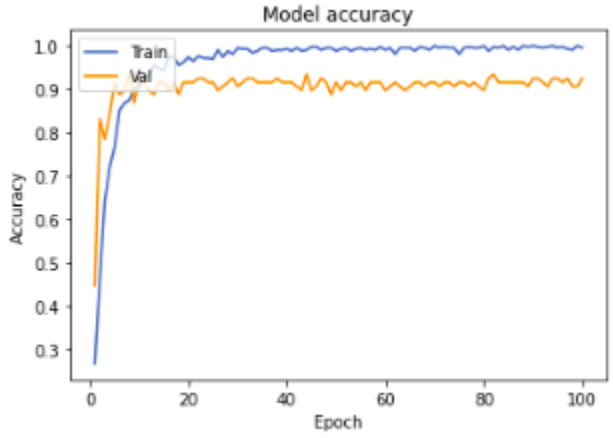

Fig 6.2 : Model Accuracy

Model Loss

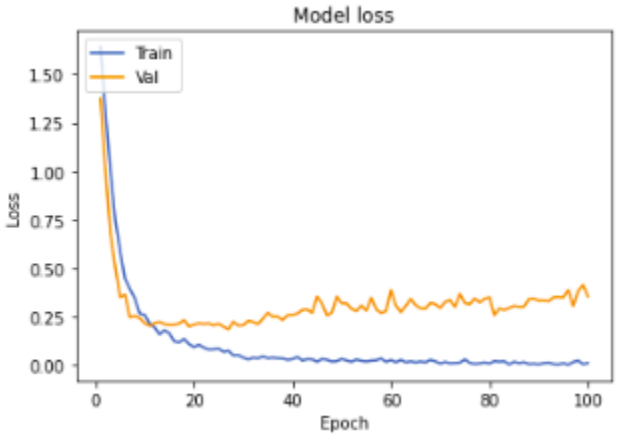

Fig 6.3 : Model Loss

6.2 Confusion Matrix

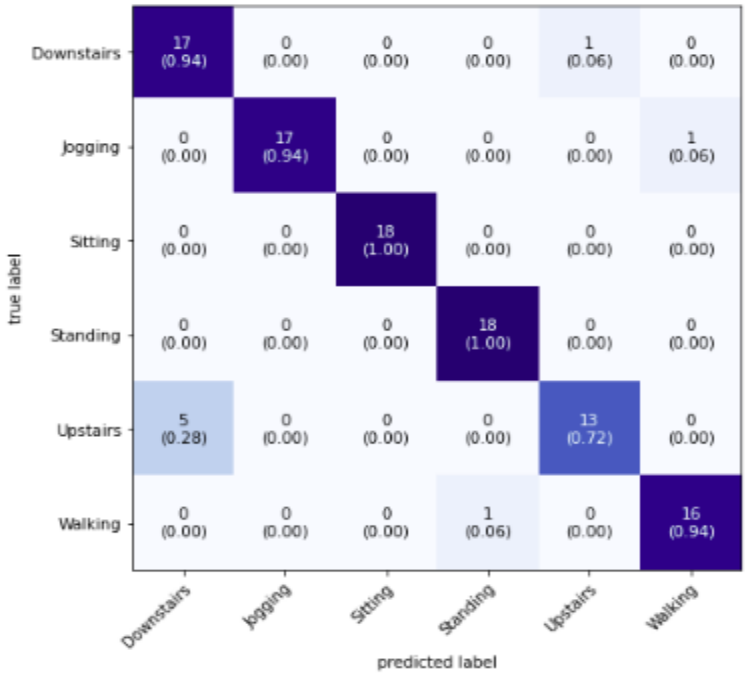

Fig 6.4 : Confusion Matrix

From the Confusion Matrix we can notice the Accuracy of sitting and standing is 100% but in other activity walking, jogging and downstairs are 94% while upstairs accuracy is 72%.

Chapter 7

7 Conclusion

In conclusion, real-time human activity recognition using IoT accelerometer sensors and Convolutional Neural Networks (CNN) is a promising approach for accurately and efficiently classifying human activities. By leveraging the data from IoT accelerometer sensors and employing CNN models, this approach enables the development of context-aware systems that can automatically recognize and respond to different human activities in real-time. It has applications in healthcare monitoring, fitness tracking, sports analysis, and assistive technologies. However, challenges such as data variability, sensor placement, noise and artifacts, real-time performance, model generalization, labeling, and power efficiency need to be addressed. Overcoming these challenges requires advanced algorithms, optimization techniques, and careful dataset collection. With further research and development, real-time human activity recognition using IoT accelerometer sensors and CNN holds significant potential to revolutionize various industries and improve the quality of life for individuals. It can contribute to personalized and context-aware support, enhancing user experiences and providing valuable insights for a wide range of applications.

Bibliography

1. S. Wang, C. Li, J. Wang, J. Huang, and S. C. Liu, "End-to-end Real-time Human Activity Recognition with Sensor-based Data and CNNs," in Proceedings of the 2018 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (UbiComp '18), pp. 1784-1792, 2018. DOI: 10.1145/3267305.3274113.

2. A. Ignatov, R. Real, J. Zhang, A. Drozdova, and P. Timonin, "Real-time Human Activity Recognition from Accelerometer Data using Convolutional Neural Networks," in Proceedings of the 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 1947-1950, 2018. DOI: 10.1109/EMBC.2018.8512634.

3. S. K. Ghosh, S. K. Ghosh, and S. Roy, "Real-Time Human Activity Recognition Using an IoT-Based Framework," in Proceedings of the 2018 International Conference on Information Communication and Signal Processing (ICICSP), pp. 21-25, 2018. DOI: 10.1109/ICICSP.2018.8632205.

4. H. Haruna, S. Q. Li, and K. Nitta, "Real-Time Human Activity Recognition using Deep Convolutional Neural Networks on Body Sensor Networks," in Proceedings of the 2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA), pp. 224-228, 2019. DOI: 10.1109/IEA.2019.8793852.

5. M. Bao, Z. Intwala, and K. A. Hua, "Real-Time Human Activity Recognition from Wearable Sensors by Sequence-Based Deep Learning," in Proceedings of the 2019 IEEE 21st International Conference on e-Health Networking, Applications and Services (Healthcom), pp. 1-6, 2019. DOI: 10.1109/HealthCom45863.2019.9009630.

6. Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. This paper proposes a real-time human activity recognition system based on convolutional neural networks (CNNs). The system uses a single accelerometer to collect data, and the CNN is used to extract features from the data and classify the activities. The system was evaluated on the UCI Human Activity Recognition Dataset, and it achieved an accuracy of 92.2%.

7. IoT-based Human Activity Recognition Models based on CNN, LSTM and GRU. This paper presents three different models for human activity recognition using IoT sensors. The models are based on convolutional neural networks (CNNs), long short-term memory (LSTMs) and gated recurrent units (GRUs). The models were evaluated on the UCI Human Activity Recognition Dataset, and they achieved accuracies of 95.8%, 96.4% and 93.6%, respectively.

8. Real-Time Human Activity Recognition in Smart Home on Embedded Equipment: New Challenges. This paper discusses the challenges of real-time human activity recognition in smart homes. The paper focuses on the challenges of multi-occupant and concurrent activities

recognition, and the challenges of dealing with the quality of the training data. The paper also discusses some possible solutions to these challenges.

9. Zhang, H., Xiao, Z., Wang, J., Li, F., & Szczerbicki, E. (2020). A Novel IoT-Perceptive Human Activity Recognition (HAR) Approach Using Multihead Convolutional Attention. IEEE Internet of Things Journal, 7(2), 1072–1080. https://doi.org/10.1109/jiot.2019.2949715

10. Mekruksavanich, S., Jitpattanakul, A., Youplao, P., & Yupapin, P. P. (2020). Enhanced Hand-Oriented Activity Recognition Based on Smartwatch Sensor Data Using LSTMs. Symmetry, 12(9), 1570. https://doi.org/10.3390/sym12091570

11. Mekruksavanich, S., & Jitpattanakul, A. (2021). Biometric User Identification Based on Human Activity Recognition Using Wearable Sensors: An Experiment Using Deep Learning Models. Electronics, 10(3), 308. https://doi.org/10.3390/electronics10030308

12. Zahin, A., Tan, L. T., & Hu, R. Q. (2019). Sensor-Based Human Activity Recognition for Smart Healthcare: A Semi-supervised Machine Learning. In Lecture Notes in Computer Science (pp. 450–472). Springer Science+Business Media. https://doi.org/10.1007/978-3-030-22971-9_39

13. Mekruksavanich, S., & Jitpattanakul, A. (2021). Biometric User Identification Based on Human Activity Recognition Using Wearable Sensors: An Experiment Using Deep Learning Models. Electronics, 10(3), 308. https://doi.org/10.3390/electronics10030308

14. Zahin, A., Tan, L. T., & Hu, R. Q. (2019c). Sensor-Based Human Activity Recognition for Smart Healthcare: A Semi-supervised Machine Learning. In Lecture Notes in Computer Science (pp. 450–472). Springer Science+Business Media. https://doi.org/10.1007/978-3-030-22971-9_39

15. Costa, K. a. P., Papa, J. P., Lisboa, C. a. V., Munoz, R., & Gupta, D. (2019). Internet of Things: A survey on machine learning-based intrusion detection approaches. Computer Networks, 151, 147–157. https://doi.org/10.1016/j.comnet.2019.01.023

16. Gomes, G., & Iano, Y. (2021). Human Movement Recognition System Using CW Doppler Radar Sensor with FFT and Convolutional Neural Network. https://doi.org/10.1109/lamc50424.2021.9602484