POWER-EFFICIENT COLLABORATIVE FALL DETECTION USING A WEARABLE DEVICE AND A ROBOT

By

RICARDO HERNANDEZ

Bachelor of Science in Computer Engineering

Oklahoma State University

Stillwater, Oklahoma

2019

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2022

POWER-EFFICIENT COLLABORATIVE FALL DETECTION USING A WEARABLE DEVICE AND A ROBOT

Thesis Approved:

Dr. Weihua Sheng

Thesis Adviser

Dr. James Stine

Dr. Bingzhe Li

ACKNOWLEDGEMENTS

I want to thank my family and friends who stood by my side during these years.

Name: RICARDO HERNANDEZ

Date of Degree: DECEMBER, 2022

Title of Study: POWER-EFFICIENT COLLABORATIVE FALL DETECTION USING A WEARABLE DEVICE AND A ROBOT

Major Field: ELECTRICAL ENGINEERING

Abstract: With increasing life expectancy, the technological sub-field of elderly care is expected to rise in importance. This growing population has put a strain on current elderly care facilities and personnel. This research work is focused on designing and implementing a collaborative elderly care system (CECAS) using a companion robot and a wearable device that will enable the monitoring and companionship needed in assisted living communities and private homes. This system is meant to improve the automation in elderly care which will lead to a reduction in costs and an increase in availability. Two elderly care applications are used to demonstrate the capability of the proposed system, which includes: fall detection and response, and conversational assistance. In the fall detection and response application, a two-step method is proposed, which consists of a motion data based preliminary detection on the wearable device and a video-based final detection on the companion robot. To further reduce the operating costs and improve the system, parameters such as the image size and the neural network size are optimized through extensive experimental study. In the conversational assistance application, the robot is capable to conversate with the user to determine if external assistance is needed and acts as an extra measure of fall detection. The results of the experiments showcase that both the wearable device and the companion robot can be optimized for power-efficiency, speed, and retain fall detection performance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The world is witnessing the rapid growth of an aging population and the number of older adults over 65 is expected to exceed 1.5 billion by 2050. This is concerning because older adults usually suffer from various health issues; for example, high blood pressure, decline of memory and cognition, mobility, etc., which calls for more innovations in the health sector sub-field of elderly care [3]. As most older adults prefer to live in their homes, there is a huge demand to support an elderly population who needs constant healthcare services but are not able to rely on themselves. On the other hand, providing a personal healthcare practitioner for each older adult is not a realistic scenario due to logistics and, primarily, financial reasons. Although many robotic technologies have been developed in recent years, how to assist older adults to live safely is still particularly challenging and a current frontier. First, it is critical if the robotic system can detect an older adult's falls and ask for help when necessary. Second, it is desirable to provide daily assistance to older adults through a conversational interface anywhere in their home. Additionally, how to avoid the privacy issue of the older adults and how to reduce the energy consumption for activity monitoring are key issues when designing wearable devices.

Companion robots are being developed as a potential solution to providing better independent living for older adults. These robots not only provide basic functions such as news, music, and daily chatting, but also offer medical services such as medication reminders, and emergency detection. Some robots could detect falls and alert caregivers for help; however, it may cause privacy concerns if cameras are used to monitor older adults' activities from a third person point-of-view. In addition, a robot, even if equipped with high quality sensors, has a limited sensing range around itself and is not able to provide sufficient monitoring capabilities in the whole home environment. Therefore, it is highly desirable to develop novel solutions to allow a robot to monitor and assist older adults everywhere in their home. One way to achieve this is by incorporating wearable technology that can assist the companion robot with acquiring necessary data. A wearable camera device could alleviate users' privacy concerns since the images being captured by it are not of the user but of the environment. In addition, a wearable motion sensor can be used to trigger the camera, thus allowing the camera to capture images only when necessary. This makes it possible to combine wearable devices and companion robots to build a wide-ranging and privacy-aware personal assistance system for older adults.

CHAPTER II

REVIEW OF LITERATURE

Several studies have been conducted to showcase that wearable devices have encouraged users to be more active and live more health-conscious lifestyles [1]. This is the case for younger individuals who are more health-conscious to begin with and intend for its effects. It is important to bring this technology to older individuals and tailor the device to their lifestyle. The beginning portion in designing a wearable device is the actual device and its configuration. Thus, it is important to understand the progression of wearable technology and other studies that have similar challenges.

2.1: WEARABLE DEVICES

Schiboni et al. [4] designed and constructed a wearable head-mounted camera setup for dietary data collection by fixing a camera on a cap's brim where the images taken were the point-of-view of the user. This configuration gives an effective point of view but does not give an ergonomic design that considers the amount of time it is expected of the user to wear it. Similarly, Sun et al. [5] developed a wearable device for health monitoring and personal assistance that could be worn on a helmet. The device consists of a powerful CPU, an image sensor, a single accelerometer, and an audio processor. It is possible to wear it on the chest which is a preferred alternative than the head-mounted design but due to the overall system size and complexity, the chest mount could cause issues and annoyance when wearing.

It is not realistic for the user to wear daily for prolonged periods. The IMMED project [14] implemented a wearable device that captures video meant for people with dementia due to age. The system is composed of a wearable camera device designed to capture audio and video data of the activities of a patient, which is later processed with multimedia indexing techniques in order to allow medical specialists to analyze several hour-long observation shots more efficiently. This study has a similar issue since the device is chest mounted using a large strap mechanism which could indirectly affect the behavior of the user. Lastly, Zhan et al. [6] proposed an activity recognition implementation that uses video data obtained from a wearable camera embedded on glasses. This is a novel and recent frontier in the wearable technology field but does pose challenges for our application. It is effective if only images are needed but any other peripherals will enlarge the design and make the overall system less user-friendly. Also, few individuals wear glasses on a daily basis so adjustment from the user will be needed.

2.1: FALL DETECTION

There are several methods that have been developed for fall detection, each with their own advantages and disadvantages. Zerrouki et al. [7] compare different machine learning algorithms for fall detection that use video sequences of different daily and fall activities as input. They compared Naïve-Bayes, k-nearest neighbors, neural network, and support vector machine (SVM) algorithms and concluded that the SVM algorithm performed best among these, with respect to most measures of performance such as accuracy, sensitivity, recall, etc. [15][16][17][18]. This study focuses on video data of activities of individuals but from a third person point of view. Similarly, Shojaei et al. [8] demonstrated an image-based detection method by using a long short-term memory (LSTM) neural network that verifies a fall with 3D joint skeleton features. These features are only possible due to the third person point of view of the individual. Both are important and relevant but are at a disadvantage since these methods would entail a device that is stationary and with a constant view of a patient.

2. 3: FALL DETECTION ON WEARABLE DEVICES

There are also several studies that have looked at possible wearable device designs for the application of fall detection and look for efficient detection methods. Casares et al. [9] implemented a resource-efficient method for fall detection by using a wearable embedded smart camera, which is a small, battery-operated unit. The proposed system uses histograms of edge orientations as well as edge strength values and analyzes their correlation to determine a fall. Their study uses the CITRIC platform, which was developed by researchers at Berkeley, originally as a device for cluster frameworks. It is reasonable in size but still larger due to the number of processing units involved since all computations are done locally. Martinez et al. [10] use a combinational neural network that consists of a convolutional neural network and LSTM to extract features from raw data such as senor data and video data for real-time fall detection. Multimodal architectures are the current frontier of fall detection systems and are common in research. To highlight this another study done by Ozcan et al. [11] also developed a fall detection system using video and accelerometer data captured by a wearable smartphone.

In summary, there are many studies throughout the years that have tackled fall detection either with or without using wearable technology but each with unique features and system configurations, each trying to determine the most efficient method. The most common methods are a stationary camera or cameras with powerful local processing units which allow for high accuracy in detection but are limited in range and do not consider the realistic lifestyle of the elderly. That is the reason for the adoption of wearable technology and its extension of functionality. For wearable fall detection, it is common to use large devices that implement several differing algorithms from simple to modern algorithms like histograms of gradients to multimodal machine learning which could either be running locally on the wearable or the data acquired sent to be processed on a central processing unit. Inspiration is drawn from these studies but due to the uniqueness of the combination of a wearable device with only a single image sensor and a companion robot, there are only a few comparisons. The choice in algorithm, wearable device and location, processing unit, and dataset are several of the parameters compared. The system in comparison is a combinational neural network running on a stationary companion robot that uses first point-of-view images taken from a small ergonomic wearable device attached to the chest for fall detection which is a unique system that has not been done. Each has been tested separately for fall detection but not in a single configuration that extends functionality of an existing companion robot with its own features. In addition, the goal is not only for the accuracy of the fall detection algorithm but to reduce power consumption which is rarely experimented on and not seen in the above studies.

CHAPTER III

HARDWARE DESIGN OF SYSTEM

The wearable device functions as a multi-sensor device that collects the data regarding the wearer and the surrounding environment. It also enables two-way communication between the user and the companion robot. The device can be broken into two main components; the hardware and the software. The hardware components consist of a microcontroller that controls and processes data, several peripherals such as an image sensor and speaker that collect or use data, the device’s circuit board that connects all electronic components, the holding container that encompasses everything stated before, and the harness which allows for the device to be wearable. Below each component will be looked at in more detail and explained why it was implemented.

In addition, another important device that is part of the collaborative system is the companion robot. Similarly, to the wearable device, there are several main components that make up the robot. Since this component was developed separately by other researchers in the ASCC Lab, only a brief explanation of the system will be given to complete an understanding of the entire system.

3.1: PLACEMENT

There were several possible positions that the wearable device would be located on the user’s body, each with their own pros and cons. The three main configurations were a necklace, headset, and magnetic clip design. The necklace design would have the wearable at the bottom portion of the necklace structure, thus located on the chest of the user but ultimately reliant on the length of the neck piece. This position gave the best point of view for the camera due to being able to capture the actual point of view of the user, however at a lower location. Surroundings and objects being interacted with by the user would be in clear view as well as the arms and legs depending on the stance of the user. This is advantageous due to the fact that key objects like the floor and ceiling can easily be learned which are paramount when detecting a fall. The main concern with this configuration is the movability of the wearable device. A string material will give no concrete position and the optimal comfort for the user while a solid neck frame will give a better possibility of consistent positioning with a drawback in comfort. The other option that was briefly looked at was a headphone type configuration. Current headphones have optimal positioning of the speaker, directly on the user’s ear, and the microphone, near the user’s mouth. No surrounding noises would cause issues in the acquisition and output of sound in this scenario. It also gives a near perfect position of the camera, closely acquiring the user’s point of view. Above it explains why this is advantageous and still applies here. Again, the disadvantage of this configuration is the ergonomics but also the realistic expectations of the user. This wearable device is to be worn for all waking hours with no interruptions except when charging. Realistically a headset would cause the user discomfort only a few hours into the day, falling short of 16 hours, the average waking period

Finally, to improve upon the original necklace configuration, a magnetic attachment was established. It will allow the user to easily take on and off the wearable device and give the position of the device some variety. It will directly be attached to the user’s clothing, recommended on the chest of the user. No strain is on the user's body, only indirectly due to the weight increase of the clothing but is minimal and spread throughout the clothing and shoulders of the user. It has the same advantages of the necklace configuration but a similar con of instability. The difference being that the stability is reliant on the user's clothing and looseness. Clothing above average in size relative to the user's frame will allow the device to more freely move while clothing tight and close to the user's frame would allow no movement of the device and will be parallel with the user's chest. To further mitigate this phenomenon, the magnetic attachments surface area was increased in size to give a more solid base along with increasing the overall amount of magnetic power between the attachment and the casing. Figure 2 shows the structure of the case with the primary part encasing the electronic components while the magnetic attachment is separate.

3.2: PERIPHERALS

The diagram, as can be seen in Fig. 1, shows the wearable consists of the main processor along with all the sensing peripherals; the processor, an ESP32-DOWD6Q chip which specializes in Wi-Fi IoT applications, a three-axis accelerometer, a digital MEMS microphone, a class D amplifier, a 2W magnet speaker, and a 4-bit GPIO expander with several other passive components not shown. The microprocessor chip is found integrated within a development board given the name ESP-CAM, which is a small, embedded computing module that provides an additional 4 MB of PSRAM on top of the chips 520 kB of SRAM but most importantly an OV2640 CMOS image sensor. This development board was also chosen for its compactness, the amount of available GPIO pins for programming, and its embedded Wi-Fi antenna. Since the older adult is expected to wear the device for a sustained period of time, a lightweight module was paramount. As for the general-purpose IO pins, there are eight available for use which each can be configured to use the I2S, I2C, and UART communication protocols, all necessary for the embedding of the above listed peripherals. Also, the antenna fitted on the development board has the capability of using the Wi-Fi standards of 802.11b, g, and n. This allows for a high data rate that takes only a few seconds at the higher end to transfer images and audio from the wearable to the companion robot. These images and audio data are large in size, up to 1MB, thus the external PSRAM that extended the chip's storage is important.

The ADXL345 digital accelerometer is adopted as the motion sensor, which is a small, ultra-low power, 3-axis accelerometer with a high resolution of 13-bits that measures up to 16g-force. The sensitivity and resolution are more than sufficient for detecting a varying degree of activities of an older individual. I2C communication protocol is used for the ADXL345 to talk to the ESP32-CAM, with a max rate of 400 kHz which allows for a sampling of 800 Hz, sufficient for the application. The audio devices are an INMP441 omnidirectional MEMS microphone and a Uxcell 2W magnet speaker driven by the MAX98357A class D amplifier. These peripherals are fitted with the necessary features to provide a high-performance, low-power, digital-input and output that uses the industry-standard I2S protocol. Finally, the embedded GPIO expander is the PCA9536DR. This component was necessary due to the ESP32 pins being fully occupied by the other peripherals and a single pin of the class D amplifier being left to be driven. It is addressed and commanded using I2C which was already in use and allowed for an easy implementation without the need of additional pins. To take advantage of the extension of the GPIOs, a button is implemented so it can be taken as an input for the user to interact with the wearable device, in this case, to instigate the recording mechanism. In addition, a blue LED is used to indicate to the user when the microphone is recording.

As shown in Fig. 1, the battery fitted to the circuit board is a lithium-ion polymer battery which provides a thin, lightweight design with a capacity of 820 mAh. The component that has a considerable amount of power draw is the ESP32-CAM, which has a maximum of 270 mA current consumption when transmitting data. The other components have a relatively small amount of power consumption, therefore only the ESP32-CAM is considered when calculating the battery life. Assuming a worst-case scenario of sixty false positives of fall detection occurring every hour, the average amount of current draw would be around 152 mA which results in a battery life of roughly 6 hours. This is sufficient for supporting daily use. Along with the physical size of the battery being 30.5mm by 43mm by 6.8mm, the battery capacity of 820mAh was chosen. The output of the battery is 3.7V at full charge. This voltage output is not compatible with the ESP32-CAM that requires a supply voltage of around 5V, therefore a voltage booster is used to increase the voltage to 5V. Along with the booster, the board is also equipped with a charging circuit that takes a USB-C connection and charges the battery at a max rate of 1000 mA. The chips to accomplish the voltage boost and charging capabilities are the TPS61090RSAR and the MCP73871, respectively. Due to the simplicity of the Adafruit Powerboost 1000c power system circuit, it was adopted and implemented in the hardware design.

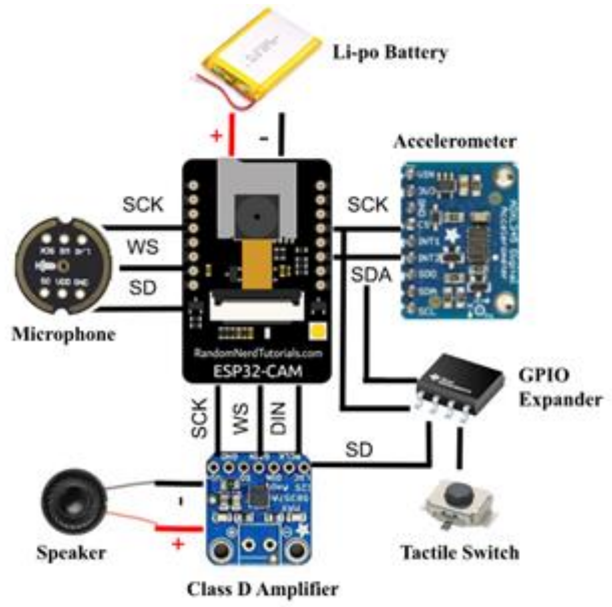

Figure 1. The design of the wearable device.

3.3: CASING

Four versions of the wearable device were prototyped and tested. This first consisted of designing a solid frame neck brace embedded with the wearable, a headphone mounted with the wearable, and various corded necklace designs. The design more suitable for older adults is a wearable device that incorporates a durable magnetic clip able to be pinned anywhere on the users' clothes, preferably on the chest where the view of the wearable is optimal. The final design integrates all the components of the wearable into one solid case shown in Fig. 4. The case utilizes two segments, the electrical component case, and the magnetic backing component, which allows for the user to pin the case on any article of clothing. The front segment of the case has three openings, one for the camera lens, the other for the microphone, and lastly for the button. There are also three other openings; one for the speaker, the charging port, and ventilation for the dissipation of heat, found on the left, bottom, and top side of the casing, respectively. As for the back segment, that is where the magnetic strips are located. These strips in combination with a separate component with four embedded magnetic discs, creates the necessary pinning mechanism. The magnetic attachment places the discs in a rectangular shape and spaces them as far as possible to give the strongest foundation with no weak points. Fig. 2 showcases the final wearable case along with the external magnetic pin component. Also, the final iteration's inside has holding positions for the printed circuit board, the lithium battery, and the speaker, and can be seen in Fig. 3.

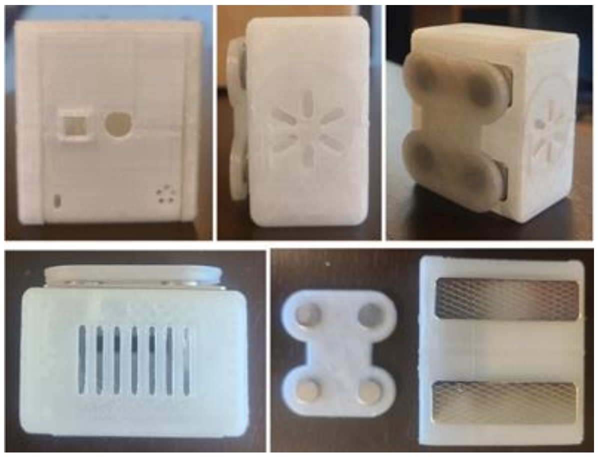

Figure 2. Front, Left, Back Left, Top, and Back of Case, with Magnetic Component exposed

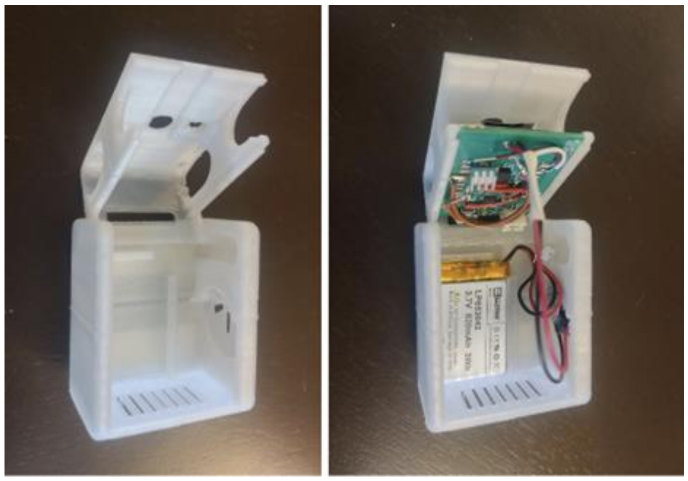

Figure 3. Inside of Case without and with the Electrical Components

3.4: PCB DESIGN

Two generations of the circuit design were constructed. The first being a design using a protoboard and several development boards. The final design is a standard two-layer printed circuit board with surface mountable devices soldered onto it. The first allowed for quick prototyping and adjustability. It guaranteed working components, and the main challenge was connecting the components correctly with each other and planning the placement of each component. Figure 4 shows the final design.

The first iteration used a single 7cmx5cm protoboard that incorporated an ADXL345 accelerometer, INMP441 microphone, PowerBoost 1000c, and ESP-CAM development board. It was larger and needed to be reduced for better ergonomics and needed to implement other components such as a speaker system, button, and LED indicators for power, and audio recording. The final design is a two-layer printable circuit board designed in Fusion360 and manufactured by JLC boards seen in Fig. 4. The design process first looked at all the development boards incorporated in the first design and some others like an Adafruit amplifier, light sensor, and GPIO expander. Any redundant components were removed and simplified to minimize the amount of components thus size. Fig. 4 shows the final schematic of the PCB. The only component kept in its original form is the ESP-CAM due to its complexity but for future work it could be an area of interest. This design allowed for a drastic decrease in size from an area of 3500 mm^2 to 1200 mm^2, a 67% decrease. Due to the limitation of two layers, some external wiring needed to be done. For future work, a four-layer board should be looked at so it can be improved, and the size can be further reduced. The bottom layer incorporates the power system and the microphone since the opening is on the bottom side of the chip and the microphone needs to be facing outward for the optimal audio acquisition. The power system, using the recommended layout found in the datasheet for the TPS61090 booster chip, has large power and ground planes to reduce impedance and current loops. On the top layer, all the other peripherals are incorporated and routed. The accelerometer has the two serial data pins being externally wired. Figure 4 below showcases the PCB design on Fusion360 along with the real-world manufactured PCB with all components soldered along with debugging wires.

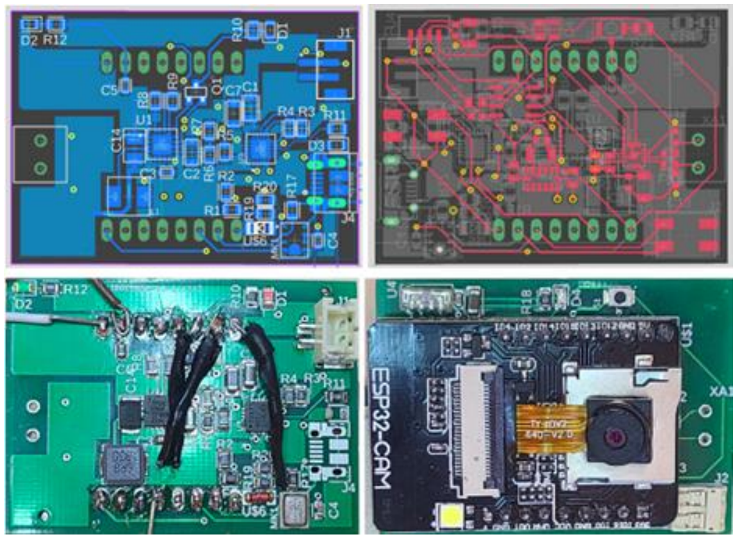

Figure 4. PCB Simulation and Real-world PCB of the Wearable

3.5: COMPANION ROBOT

As can be seen in Fig. 5, the Companion Robot [12] developed in the ASCC lab consists of three main parts; the head for the main sensor system, the body for structural support and housing, and the base where the main power unit resides. The robot head incorporates the Intel

Realsense RGB-D camera, four microphones for speech recognition and sound localization, and a touch panel powered by an ARM-based board running an Android OS for the user to interface with the overall system. The body houses an Intel NUC with an i5 processor which is the main computational unit that runs the speech recognition, video-based fall recognition, and several other features that provide a hospitable companion for the older adult.

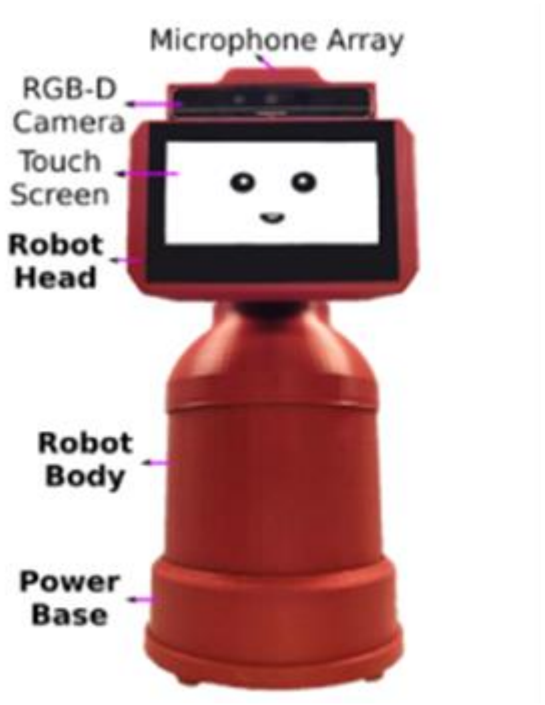

Figure 5. Companion Robot Diagram

CHAPTER IV

SOFTWARE DESIGN

The software design of the wearable can be divided into two parts: the wearable device part and the companion robot part. For the wearable device, the main task is to trigger the camera for data transfer to the companion robot and to communicate audio between the two devices if needed. There are several peripherals working in tandem to accomplish this flow, which requires software designs for optimization and speed. As for the companion robot, data communication and processing are essential to the working of the entire system, which includes the fall detection algorithm.

4.1: DUAL-CORE THREADING

To fully take advantage of the ESP-CAM capabilities, the flow of the program was broken into several tasks that could be pinned to specific cores for full utilization, reduction in latency, and parallelization. The four main tasks are the Wi-Fi, accelerometer, camera, and speaker and microphone. Having the Wi-Fi on a separate task allows for the parallelization of communication between the wearable device and companion robot and data acquisition. This ensures that detection of events such as a fall, will not be interrupted by data transfers of other tasks or even the accelerometer task. Similarly, the accelerometer and the other peripherals are separated so that an event like a fall is captured and not ignored due to another task running.

Figure 6 displays the tasks and their respective cores. The accelerometer and the camera are paired together on a single core since most times that a camera event is triggered is when the accelerometer passes the set threshold for a fall. There will be a slight decrease in sampling speed for the accelerometer but only affected for less than a second when the camera finishes capturing the images. The other tasks on the other cores will be running in parallel and should not affect the other two.

Communication between the tasks is done using queues when large bytes of data are being transferred and notifications when only a trigger needs to be communicated. For example, only a notification is used to communicate from the accelerometer task to the camera task that a fall has occurred, and ten images need to be captured. This method, stated in the documentation of FreeRTOS, uses less clock cycles than queues thus reducing the latency of the program [22]. Figure 6 highlights the full communication network between the tasks. Along with the queues and notifications, there are specific commands set to better distinguish the intentions of the data being transferred. Notifications to the microphone and speaker task have specific bits to indicate which of the two functions is to be called. Similarly, the camera task uses a notification system to indicate to take images due to a fall.

4.1.2: TASKS

As stated before there are four main tasks involved in the wearable device. The program was separated into tasks to take advantage of parallel processing and reduce latency in fall detection. Having several tasks that have clear functionality reduces the overall complexity and increases the readability. It also allows for parallelization and constant monitoring of the acceleration of the user and for quicker data transmission to the companion robot. The accelerometer task has a function to initialize the I2C driver to communicate with the ADXL345 board, constantly monitor the magnitude of acceleration and determine if it has passed a preset threshold, and if so, as its final function, trigger a notification that is sent to the camera task. As for the camera task, the first function is to initialize the camera driver which is using a serial camera communication bus (SCCB) driver. At initialization of the task, it waits for the camera configuration data that is expected from the Wi-Fi task. From this data the image resolution is determined, and an input was created for the driver to configure the camera capture resolution. Once this is done, the task waits for any notifications that will determine if the set IMAGE_NUM (by default set to 10) number of images are to be captured.

The next task is the combined task of the microphone and speaker. The main reason for this combination is because only one channel is available for I2S communication, the other being used by the SCCB driver of the camera. This means that the microphone and the speaker cannot both be used simultaneously. This is not an issue since only one of these should be functioning at a time realistically. In the worst-case scenario, they should work back-to-back for the companion robot conversation functionality. Thus, the driver needs to be reinitialized each time the microphone or speaker function is triggered and de-initialized after the function reaches its end. The only initialization that is seen in the beginning of the task is for the GPIO expander. The expander allows for the monitoring of the pressing of a button which will trigger the microphone to record for the set time of RECORD_SIZE (by default 7 seconds). The task also waits for a notification from the Wi-Fi task that is used to determine if a response is needed, thus a recording of the user, or that speaker data is being transferred, thus triggering the speaker functionality.

Lastly but most importantly is the Wi-Fi task. It uses a basic TCP socket connection to transmit and receive data. To begin, the socket is waiting for the camera configuration data to be transferred from the server. This is sent by queue to the camera task for initialization. It then enters its main loop where it has three main blocking functions. The first is the socket receive function that waits on a command from the server. This command determines whether there is speaker data received and sent by queue to its respective task, to trigger the recording of audio, or to retake an image through notifications. The other two blocking functions are two queues that wait on data from the camera or microphone task.

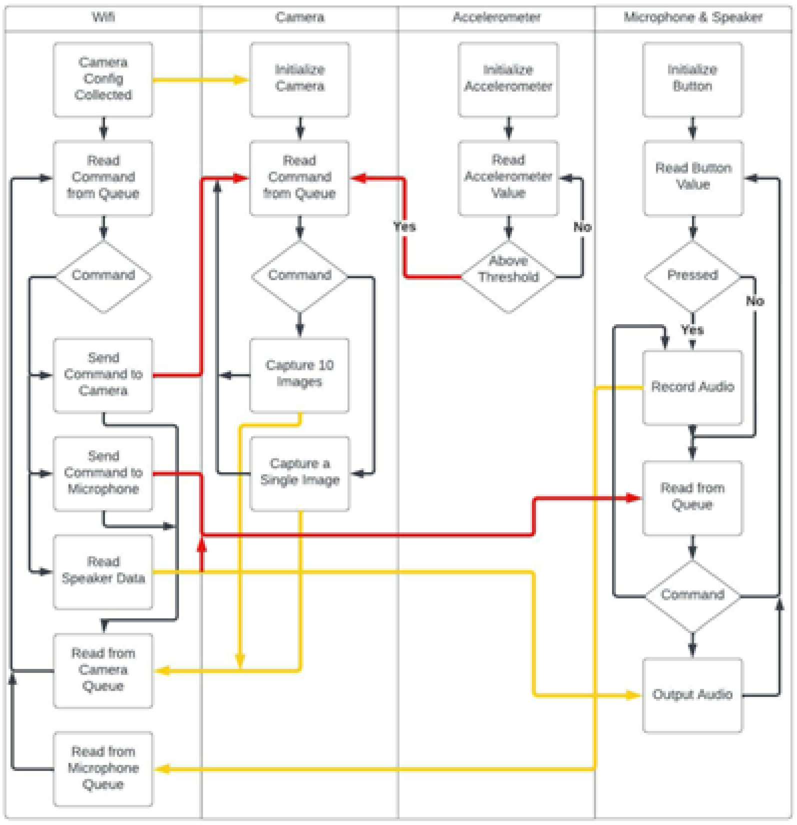

Figure 6. Wearable Software Flowchart

4.2: COMPANION ROBOT

The main task of the companion robot is to act as a server and interpret the data from the wearable device. Currently it waits for any “tags” from the wearable to tell the robot what type of data is about to be sent. This is important because it determines the type of file to save the data as and what flow in the algorithm it should take next. Clearly if images are sent then the course of the algorithm is to save the images as JPEGs and to run the model using the newly saved images. The output is a byte that either states it was a fall or not. If so, the robot tries to confirm with the user if any assistance is needed or if it was not a serious fall that necessitates any emergency services. It also acts to mitigate false positives. This confirmation is done by sending the wearable a command that tells the device to record and send the data to the robot, where speech-to-text along with text recognition is used. The conversational flow uses the same techniques when transferring data to and from the wearable but in the case of sending audio data for the speaker, a specific command is sent to the wearable that the device understands that it will be receiving audio data for output. Figure 7 illustrates the overall system flow of data and the systems capabilities.

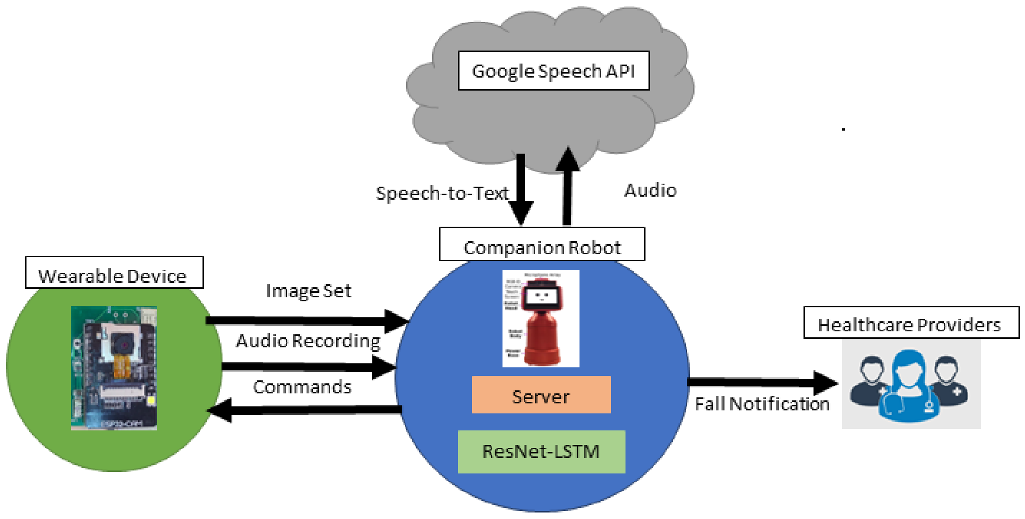

Figure 7. System Diagram

CHAPTER V

COLLABORATIVE FALL DETECTION

Fall detection relies on both the wearable device and the companion robot. However, most of the processing is done on the latter with the former communicating the necessary information. This is because the processing algorithm uses a neural network with several thousand parameters, requiring an exceptional amount of memory space. There are several other methods of detecting a fall such as edge detection, histograms of gradients (HOGs), hidden Markov model, and support vector machines etc., but these systems require tenuous tuning and designing to differentiate a fall. It would require even further development and experience to differentiate other activities. Due to the extensive research and feasibility of neural networks, it was chosen as the method that would be used for this particular application.

For image data, a convolutional neural network (CNN) was seen as the optimal architecture for feature extraction. To further improve the system, several images would be used, which requires the integration of a recurrent neural network (RNN), a sequence classification architecture. The combination of the two has the strengths of both systems and gives a robust architecture for the recognition of a handful of activities.

5.1: INTRODUCTION TO CNN

A CNN or convolutional neural network is a deep learning algorithm that takes in an array of data, most commonly an image, and assigns value, through learnable weights and biases, to certain regions or shapes and is able to differentiate between them [21]. The design of the CNN is influenced by the visual cortex of the human brain or neurons that activate due to stimuli in specific regions of the vision field-of-view. The purpose of a CNN is the ability to capture spatial and temporal relationships of data or an image and the main building block to accomplish this is the kernel or filter that is used to convolve over the image.

The function of the convolution of the image is to extract features such as edges, color, gradient orientation, etc. from the image. In the beginning, low-level features such as the ones stated above are extracted and as layers of convolution are added and learned, high-level features are extracted such as shapes, region importance, etc. which in the end gives a whole and complete understanding of the dataset which is similar to our human brains. Another fundamental building block of a CNN is the pooling layer. It has a similar function of reducing the spatial size of the convolved feature or output of a convolution layer. The most commonly used is a max pooling layer which extracts the most dominant features while not affecting its position. It takes the max value within the same portion covered by the kernel.

Finally, the output of the convolutional neural network is flattened and fed into a feed forward neural network or commonly known as a fully connected layer. This allows for the learning of non-linear combinations of the high- and low-level features learned from the CNN. Ultimately it learns which features are dominant and distinguish them throughout the dataset and classify the input after being fed through a SoftMax function. The overall operation of a convolutional neural network is demonstrated by Fig. 8.

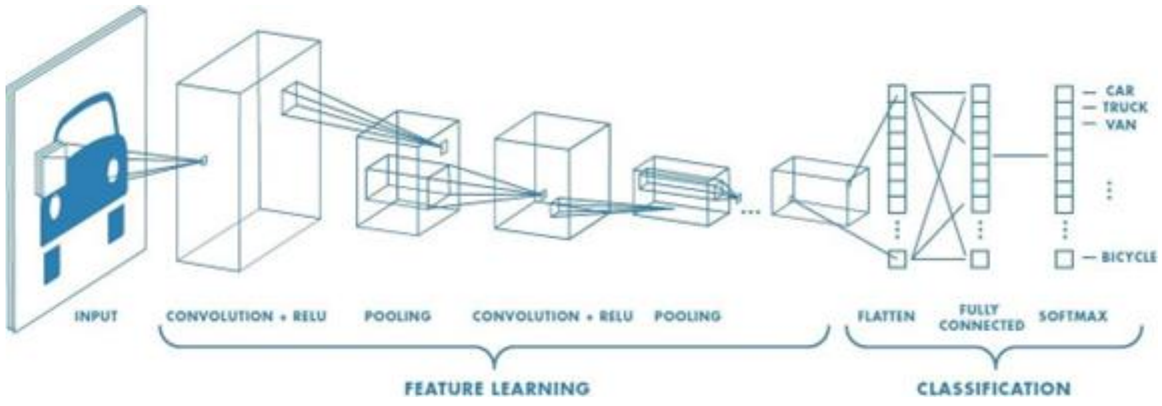

Figure 8. Convolutional Neural Network Diagram

5.2: INTRODUCTION TO LSTM

To explain the functionality and necessity of a long short-term memory neural network, first a recurrent neural network (RNN) needs to be introduced. An RNN is used for sequential learning; classification, labeling, and generation. For brevity only sequential classification will be discussed. In this scenario, only a single output is needed and dependent on every input. Similarly, to a feed forward neural network, the input is fed through a function made up of weights and biases, but uniquely also along with the current input, the previous functions output is fed in as well. This is built up until the end of the sequence and finally fed into a SoftMax function for classification.

The issue with this model is in the learning process. The loss function used to correct the error of the model is used in every input sequence. In a long enough sequence or small enough error, a phenomenon known as the vanishing gradient problem occurs. This issue influenced the creation of the long short-term memory model. It is designed to efficiently retain useful information throughout the full sequence. The LSTM accomplishes this using a series of “gates” which influence how a sequence of data comes into, is stored, and output throughout the network [20]. There are three main types of gates found in a LSTM; a forget gate, input gate, and output gate. These gates can be seen as filters and are their own neural networks. Fig. 9 is a visual representation of a single LSTM component and displays each type of gate and its functions.

The first stage of the LSTM is the forget gate. Here is where it is decided which bits of the cell state, or long-term memory of the network, are useful using the previous hidden state and the current input data. This is done by taking both as inputs into a neural network with a sigmoid activation function at its output layer; this ensures that the output is a value between 0 and 1. Zero indicates that the current data is irrelevant and the closer to one, the more cell state data is passed through. It acts as a sort of filter for long term memory. The next step involves the new filtered memory and the input data. The importance of this step is to determine what new data should be added to the long-term memory, or cell state, of the network using the previous hidden state and the input data. There are two main components to this step; the update and input neural network, which both take the previous hidden state and current data as input. The update component learns to generate an update vector, essentially containing data from the current input data in the context of the previous hidden state. This vector updates each point in the long-term memory. This is not entirely true because which input data is relevant needs to be determined and this is where the input component comes into play. Like before in the forget stage, the input gate filters the data to determine its relevance, giving a value between zero and one. This new data is then multiplied by the update vector which gives the correct components that need updating. This final output is then added to the output cell state of the forget stage. The last step is the output gate component, developing the new hidden state. The cell state cannot be used since it is all the relevant information of the sequence up to that point while only a general prediction of the next state of the sequence is needed. Again, similarly to the forget gate, the input and previous hidden state are fed through a filter to determine relevance, then multiplied with the updated cell state to retain the useful information.

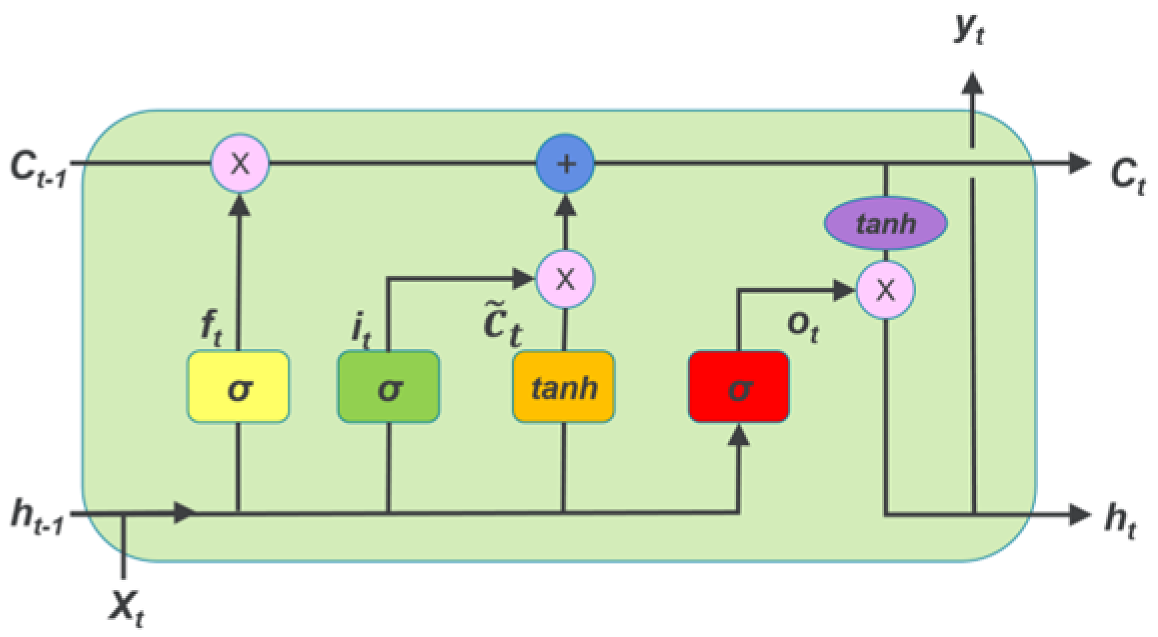

Figure 9. Long Short-Term Memory Diagram

5.3: CNN-LSTM

A limitation of a long short-term memory network is that input needs to be in the format of a vector. An image could be flattened into a vector, but any spatial or neighboring pixel dependency is lost or would be challenging for the network to learn. A CNN-LSTM was designed specifically to overcome this limitation and allows for the sequence classification of spatial inputs, such as images or videos. The first stage is the convolutional neural network that extracts the relevant features of the dataset followed by the long short-term memory network that can intake a sequence of these feature vectors and determine a classification. This architecture is appropriate for problems that have spatial input structures such as the 2-dimensional structure of pixels of images and which also have temporal structures such as the order of images in a video. It is a simple architecture but effective.

5.4: RESNET-152

From above it is apparent that the LSTM is heavily reliant on CNN and its extracted feature vectors. A CNN that is ineffective in capturing the relevant features of a dataset will always create an ineffective CNN-LSTM model no matter the effectiveness of the LSTM. To increase the effectiveness of the CNN, intuitively, requires increasing the depth or amount of convolutional layers but similarly to the LSTM, the problem of the exploding or vanishing gradients occurs. An ingenious solution was introduced to mitigate this issue with the design of the skip/shortcut connection. Simply it adds the input of a block of convolutional layers to the end of the block or output. These convolutional layers are learning a residual mapping. Even if some vanishing of the gradients occurs within the weights, the identity of the input is there to fall back on.

There are three methods in using the skip/shortcut connections which is because in some cases the input is smaller than the output dimensions. The first performs identity mapping with zero padding to increase the dimension for all shortcuts. The second has a mixture of the above method and the use of the projection shortcut where the input is bypassed but is first passed through a convolutional operation. This increases the number of overall parameters to be learned. The final method is where all shortcuts use projection shortcuts. Another key component of the ResNet architecture is the implementation of the bottleneck design. One by one convolutional layers are added to the beginning and end of a block which has been studied to reduce the number of parameters without damaging the performance of the overall network.

All these components working in tandem allows for the creation of the 152-layer ResNet architecture. Compared to other state of the art architectures such as VGG-16, GoogLeNet, and PReLu-net, it outperformed them with a top five error rate of only 5.71% with the next best error of 7.38% from PReLu-net [19].

5.5: RESEARCHED ARCHITECTURE

Taking all of this into account, the advantages and disadvantages, the system that was implemented for our collaboration system for fall detection is a combination of a ResNet-152

CNN and a LSTM developed by HHTeng [13]. This architecture was originally developed for action recognition within a video sequence in mind. It achieved a high accuracy of 85.68% using a pretrained ResNet-152 CNN based on the image dataset ILSVRC-2012-CLS, that contains 1000 plus different objects such as animals, household items, and more. The dataset in total contains over a million sample images, each with the dimension of 469 by 387 pixels but then usually preprocessed to the size of 256 by 256 or 224 by 224. In this implementation, the author chose to resize the images to 128 by 128. This architecture differs from the original ResNet by replacing the final feed forward fully connected layer with only a simple two layer fully connected layer that is configurable in number of hidden nodes but by default set to 512 and 256, respectively. The LSTM is a generic three-layer neural network that is dependent on the output of the ResNet-152 CNN.

CHAPTER VI

EXPERIMENTS AND RESULTS

In this chapter the experiments undertaken to complete this system are described in detail followed by the results obtained.

6.1.1: IMAGE DATA-COLLECTION

For deep learning to be effective, a strong training set is needed with accurate labeling. Unfortunately, there are not any recognized datasets that contain point-of-view images or videos of a person falling or even more common human activities such as walking or eating. This limitation is one challenge that has an enormous effect on the development of this system but hopefully this work can start an interest in developing such a dataset.

To begin with, data collection of various falls was conducted. The system was developed to capture ten images after passing a preset threshold of 1.35 m/s^2. This number is determined after various experiments and is the best value in triggering the algorithm for almost all falls but also whenever a person is to sit or stand. These images are then sent over Wi-Fi to the server for saving and organization into their respective folders; the folders each containing only one sort of activity such as falling or sitting. Each fall is further separated into four differing falls: falling forward, falling backward, falling to the right, and falling to the left. This is due to the fact that each entails a large differing point of view of the user, each with their own unique characteristics.

For example, falling backward will most of the time display the ceiling in its entirety, taking up the majority of the image. For the others, only a fraction of the image will contain a ceiling, with falling forward much of the time having no indication of the ceiling, in fact, the opposite, the floor. The other three activities that were captured are sitting, standing, and walking. These activities encompass practically all other movements that could occur when an elderly individual is wearing the device. Sitting and standing are intricately linked since the images ideally will be identical but in reverse order. Sitting is a more interesting activity because a quick and low enough sit could be similar to a fall, giving a false positive.

Below is a figure of the data collection environment. It was conducted in the mock smart home of the Laboratory for Advanced Sensing, Computation, and Control (ASCC Lab). It includes a mock kitchen, a dining room, a living room with an office computer, a bedroom, and a bathroom. All tests were restricted to the living room, but this does not limit the variety of images captured. For the fall actions, ten differing positions and angles were used within the living room, each surrounding a safety mattress located on the floor of the living room. Figure 11 is a top-down view of the environment and indicates with red dots the ten fall positions. The mattress allowed for the capture of realistic free falls with minimal risk of injury of the experimenter. These ten angles each have differing views of the mock home. The goal was to attain scenarios that would be common in most falls; one where only a wall is seen, only a single object such as a chair is seen, multiple objects with varying complexity, and an entirely open area.

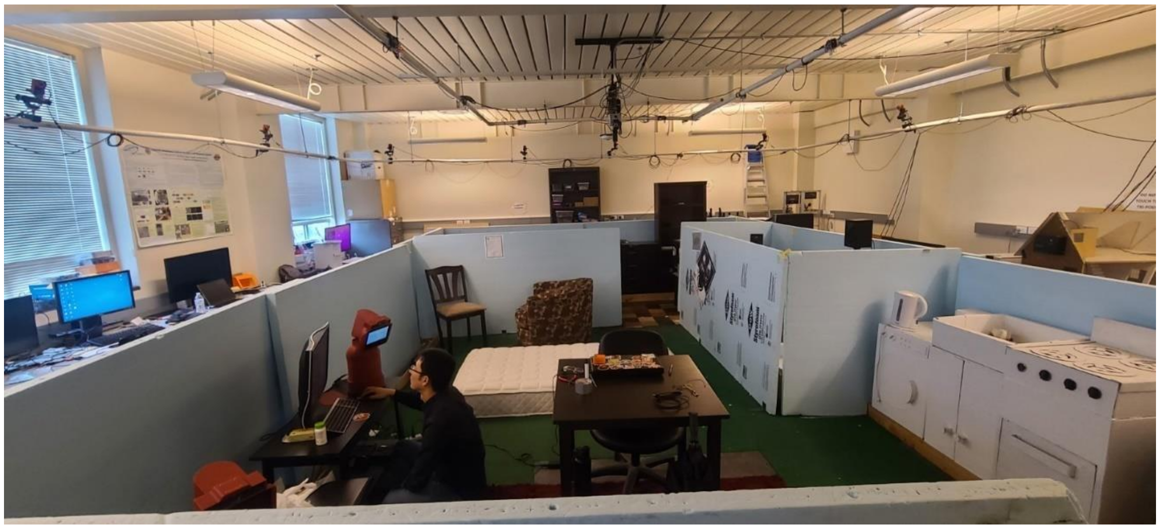

Figure 10. ASCC Mock Home

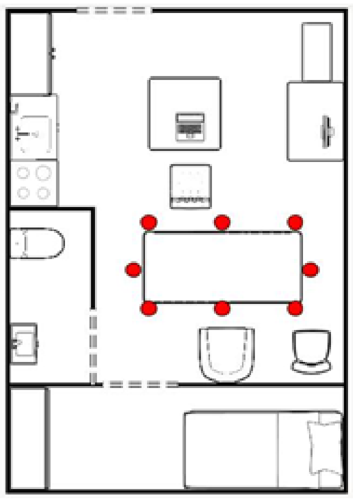

Figure 11. Fall Points in the Test Environment

For sitting and standing, there was a different method in the collection of samples. A single position of a chair was used in the dining room with varying facing positions along with two other chairs in the living room. The chair in the center of the south wall has three angles for sitting and standing while the other corner chair only has two. This in total amounts to 13 uniquely different scenarios for sitting and standing. Fig. 12 demonstrates the positions used for standing and sitting.

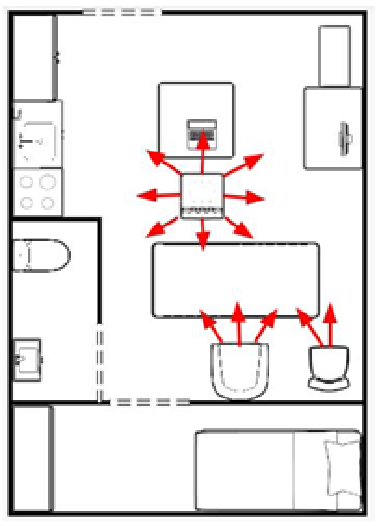

Figure 12. Sitting and Standing Viewpoints

Finally for the walking activity, there are seven walking lines in total. Each line also can be traversed in two directions, thus a total of 14 varying scenarios. Fig. 13 demonstrates the seven lines found in the mock home.

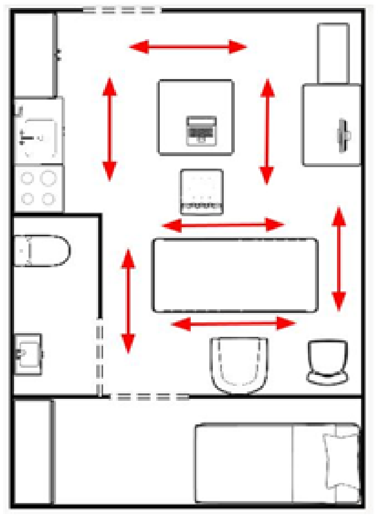

Figure 13. Walking Paths taken in the Test Environment

Due to time and resources limitations, we collected 247 samples. As can be seen above, the variety in samples was a crucial factor to best replicate reality and the unpredictability of a user's point of view in each activity. Figures 14, 15, 16, & 17 are a few examples of the image sets taken by the wearable device. The first two demonstrate the similarity between the sitting and standing samples. The beginning of both samples are quite similar since the action entails the user to first bend over partially to load the legs and the difference mainly comes from the fact that the end point of view is higher or lower, depending on if the user is standing or sitting. The latter two figures are an example of the complexity of the samples due to the volatility of the user’s chest movements and external objects that could obstruct the majority of the image pixels. The combinatorial neural network will have to learn features distinctive to its classification such as the movement of the object within the frames and the ending horizon which gives hint if a fall was to the right or left. This is expected of the neural network, but the ingeniousness of these architectures is its ability to extract features that humans look over.

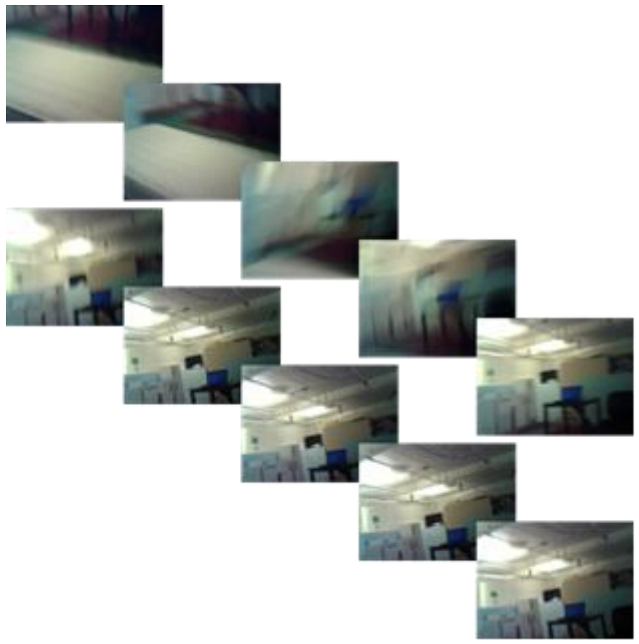

Figure 14. Sitting Image Set

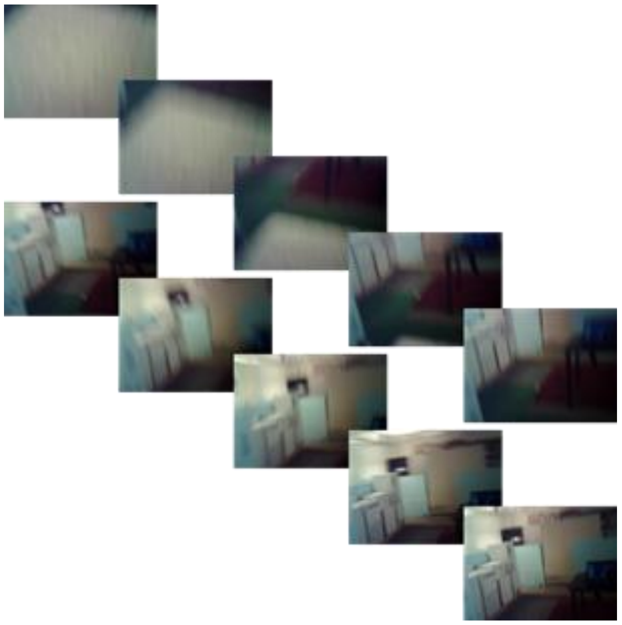

Figure 15. Standing Image Set

Figure 16. Falling to the Left Image Set

Figure 17. Falling to the Right Image Set

6.1.2: CNN-LSTM TRAINING

Once all the data was collected, formatted, and organized, the training of the model begins. The main tool used is Google Colab which allows for the use of a high-performance GPU and for easier compilation of the code without having to worry about Python versions and libraries. The main dataset of 247 samples, each in the resolution of SVGA, were uploaded to GitHub and using Google Colab functionality, this repository was able to be downloaded and used. As stated before, the main components of research are the resolution, ResNet fully connected layer parameters, and the LSTM parameters, as well as the learning rate which was optimized for the highest accuracy. The two fully connected layers had their values set to 640 and 480, respectively. This value differs from the original authors' set values but was used in other experiments in the ASCC Lab, so it is labeled as the default value. As for the ResNet output dimension, it was set to 512 thus the feature vector of the model is of this size. This in turn makes the input size of the LSTM equal in dimensions as well as the hidden layers node size. Finally, the fully connected layer of the LSTM was set to half the size, thus 256 nodes. After several runs and experiments, other parameters were set to default values for all following runs so as to be consistent and keep the experimentation less complex than it needed to be. It was determined that the batch size that gave the best results in the beginning runs was 45, the amount of epochs set to 50 since it was apparent the highest accuracy was attained before this time, and finally the number of hidden layers of the LSTM is kept constant to three which is the original value.

For every run, three main resolutions were tested; the default resolution that the model was originally optimized for, 128 by 128 pixels, 256 by 256, and finally a resolution halves the size of the default value, 64 by 64. These values encompass a large enough range to have a drastic effect on the accuracy and performance of the system from the wearable device camera acquisition, data transfer, and companion robot classification. The first experiment was to determine the best value for the learning rate. The two values tested were 1e-3, the default value, and 1e-4, two common values used by neural networks. Once the learning rate that gave the better results overall was determined, it was set as the default value for the following experiments. The next experiment varied the number of hidden nodes of the two fully connected layers of the ResNet-152 model. As stated before, the default values are 640 and 480 but the original values were set to 1024 and 768. This set of values was tested to further prove that the former values give a higher accuracy. To push the model and experiment further, another set of values were tested and compared; a fully connected architecture of 320 and 240, a direct halving of the default values. The final experiment of the ResNet-LSTM model looked at the effects of the change in the final dimension of the ResNet which feeds into the LSTM. This in turn also varies the input size of the LSTM and the fully connected layers architecture. The default value of 512 was tested and compared with a lower value of 256. The tested value of 256 changed the first layer of the fully connected component to the same value and to keep the same ratio of the default architecture, the second layer was set to 126. To further the experiments and to have a deeper look at the effects of the change in parameters of the model, this experiment was done twice with the default value of the ResNet fully connected layers, and the newer set of 320 and 240 hidden nodes.

The value that is being used to determine the original performance of the architectures is the accuracy of the validation dataset or 25% of the original 247 sample dataset after training the model. Using these accuracy values, the best performing is chosen for each resolution and further tested for its performance. From before, the model parameters or pth files that contain all the weights and biases necessary for the model are saved. These are used to upload the model and tested for its accuracy for three different sized datasets; the 25% validation dataset, the remaining samples dataset, and the full original dataset of 247 samples. The overall accuracy, the accuracy for each class, and a confusion matrix are the goals of this testing. The confusion matrix is constructed by setting positives to whenever any of the four types of falls is classified and any other activity, such as sitting, standing, and walking, gives a falsehood.

6.1.3: ENERGY CONSUMPTION

Due to time constraints and a performance degradation of implementing a bilinear image down sampling algorithm on the ESP-CAM, the resolutions closest to those tested for the ResNet-LSTM, that are provided by the camera driver, were used. These resolutions are 96x96, 160x120, and 320x240 pixels. Compared to the tested resolutions, the difference for each is 55%, 17%, and 15%, respectively.

To measure the power consumption, a Ruideng USB tester was used in tandem with the USB to TTL converter that helps communicate and flash the ESP-CAM. The converter dissipates some amount of energy itself but the difference in energy consumption at runtime and Wi-Fi transmission is what is important thus eliminating the effect. The Ruideng USB tester is a development tool that measures the real-time voltage, and current of the device that it is attached to. The wearable is then configured with its necessary resolution and constantly triggered to capture images. The current during the transfer of images over Wi-Fi is then measured and captured. Along with this data, the number of milliseconds needed to transfer a single image is measured on the ESP-CAM, captured, and averaged to determine the average time to transfer the set IMAGE_NUM or 10 images in this case. With these two values, the amount of energy consumption needed to transfer a set resolution of images can be interpreted.

To see more of a real-world application, other resolutions were also tested since the above resolutions would not be able to be resized to the three original image sizes without stretching or compression, thus loss in image integrity. The latter two were replaced with image resolutions of 176x144 and 400x296, respectively. These resolutions would be able to be used without resizing on the ESP-CAM and could be sent to the robot for fall detection with resizing being done at this point in the system.

6.2: RESULTS

6.2.1: RESNET-LSTM

As stated in the previous chapter, the first step in the experimentation of the ResNet-LSTM is the learning rate. The below figure has the results for the learning rate of 1e-3 and 1e-4, with all other parameters set to their default values.

Table 1

Learning Rate Experiment

Resolution	1e-3	1e-4
64x64	74.47%	**79.79%**
128x128	76.6%	**86.17%**
256x256	78.72%	**85.11%**

From the table, for all resolutions, the learning rate of 1e-4 consistently resulted in a higher accuracy. Due to this fact, the default learning rate for all further experiments was set to this value of 1e-4. As a side note, an interesting observation is that the increase in resolution for the value of 1e-4 did not see an increase in the accuracy which is not the case for the default value.

For the next run, the ResNet fully connected layers were experimented on. The following table highlights the results for 320 and 240, 640 and 480, and 1024 and 768, with the default learning rate value of 1e-4.

Table 2

ResNet Fully Connected Layer Hidden Nodes Experiment

Resolution	320 & 240	640 & 480	1024 & 768
64x64	80.85%	79.79%	74.74%
128x128	86.17%	86.17%	84.04%
256x256	82.98%	85.11%	82.98%

The first observation that is important to discuss is the ineffectiveness of increasing the number of fully connected nodes of the hidden layers from either 320 and 240 or 640 and 480 to the values of 1024 and 768. It would intuitively be expected that the accuracy for the latter values be higher, but it does worse in every resolution except for the 256x256 pixel resolution, where it ties in accuracy with the set values of 320 and 240. Another interesting observation of the results is that the former two values of the fully connected nodes do not have a clear advantage over the other. Both tie in accuracy for the default resolution of 128, the smaller of the set does better for the smallest resolution of 64x64 pixels, albeit by only 1.06%, but does worse for the larger resolution by a larger margin of 2.13%.

Taking the above comparison into account, both values are used for the following experiment to get an in-depth comparison and understanding of the change in parameters. The first set used was the default values, since it only did worse by only a small margin, thus the expectation was that it would do better overall. The table for the results is shown below. As described before, the final dimension of the ResNet-152 architecture was varied with the learning rate set to 1e-4.

Table 3

Final Feature Vector Dimension Experiment w/ 320 & 240

Resolution	512	256
64x64	79.79%	78.72%
128x128	86.17%	86.17%
256x256	85.11%	89.36%

There is an increase in accuracy for the highest of the resolutions but similar or slightly worse results for the other two resolutions. With further testing, the gap in the lowest resolution accuracy could be minimized or result in a higher accuracy for the lower final dimension of 256. With this in mind, the lower final dimension did improve the accuracy or did a comparable performance in classifying the seven activities.

Next the other set of values for the fully connected layers was set as the value, with the final dimension being varied. The table below showcases the results of the experiment. Again, to clarify, the learning rate is consistent and set to 1e-4.

Table 4

Final Feature Vector Dimension Experiment w/ 640 & 480

Resolution	512	256
64x64	80.85%	84.04%
128x128	86.17%	85.11%
256x256	82.98%	92.55%

This table is comparing an architecture designed with the fully connected layers hidden nodes set to 320 and 240 respectively and varying the resolution of the input images and the effects of changing the default final dimension of the ResNet-152 from 512 to 256, which invariably changes the structure of the LSTM; the amount of hidden and fully connected layer nodes. As can be seen all the parameters are reduced in this portion of the experimentation thus it is interesting to see it has the greatest effectiveness to classify each activity for both the resolutions of 64x64 and 256x256 pixels. The default resolution of 126x126 had a slight decrease in accuracy, 1.06% to be exact, but again, another few runs of testing could see a diminishing in this gap.

Looking at specifically the default resolution of 126x126 and all its accuracies throughout the experiments, the highest it could achieve with a wide range in variation of the architecture is only 86.17%. It appears there is a limitation that was quickly reached for this resolution and only further alterations, possibly within the ResNet-152 architecture, would influence this ceiling.

From the above results, the best achieving architecture for each resolution was chosen for the following experiments: the 84.04% and 92.55% accuracy architectures for the 64x64 and 256x256 resolutions, respectively and for the 128x128 resolution, there are several options, but the smallest of the models was chosen. As stated previously, the architectures are tested for accuracy on three different sized datasets. Table 5 shows the results.

Table 5

Accuracy using different sized datasets

Resolution	Validation	Training	Full
64x64	93.55%	95.14%	94.74%
128x128	96.77%	93.51%	94.33%
256x256	98.39%	97.30%	97.57%

As expected, the largest dataset does not give the highest score for accuracy but is the most important since every available sample is tested against. Interestingly, the architecture for the smaller 64x64 resolution outperformed the larger resolution, 128x128.

Below the accuracy score is broken down into their respective classification. The disparity for the middle resolution can be easily seen here. It has the most difficulty determining if a fall to the left or right has occurred but makes up in performance in other classifications to level its overall score to a similar accuracy as the lowest resolution. Unsurprisingly, the lowest resolution architecture does not excel in every classification but overall has high scores and the highest resolution excels in most classifications.

Table 6

Accuracies of Each Classification on Full Dataset

Res	FallBack	FallFwd	FallRight	FallLeft	Sitting	Standing	Walking
64x64	94.29%	97.09%	94.12%	93.33%	100%	91.67%	93.02%
128x128	100%	97.06%	85.29%	80%	100%	94.44%	100%
256x256	100%	100%	91.18%	96.67%	97.14%	97.22%	100%

The confusion matrices for the resolutions are given below. It helps to show that the 128x128 resolution architecture does the worst in determining that a fall has occurred and determines a fall the most when it was not but is comparable in performance when determining the other non-fall activities with the highest resolution architecture. The 64x64 architecture is in the middle in performance for falls but is the worst in determining non-fall classification. A positive is classified as a fall and a non-fall as a negative.

Table 7

Confusion Matrix for 64x64 Resolution

	Actual Positive	Actual Negative
Predicated Positive	126	108
Predicted Negative	8	5

Table 8

Confusion Matrix for 128x128 Resolution

	Actual Positive	Actual Negative
Predicated Positive	121	112
Predicted Negative	12	2

Table 9

Confusion Matrix for 256x256 Resolution

	Actual Positive	Actual Negative
Predicated Positive	129	112
Predicted Negative	4	2

6.2.2: ENERGY CONSUMPTION

In the following section, two sets of experiments were conducted, the first testing resolutions that result in the closest size in bytes to the ResNet-LSTM resolutions to showcase the difference in energy consumption if some pre-processing of the images was done before transferring the images to the server, and the second, a set of sizes that could be used realistically with resizing occurring on the robot. For each graph below, for power consumption, and transmission time, 100 samples were used. The graph below gives the values measured during the first experiment.

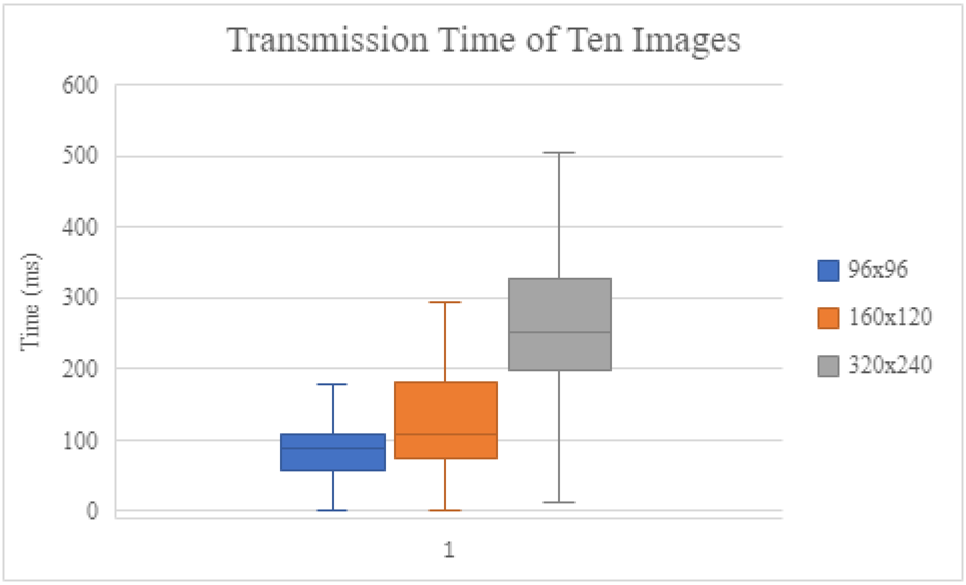

Figure 18. Transmission Time of Ten Images for First Set of Resolutions

The results measured are exactly what was expected since the resolutions are doubling from 160x120 to 320x240 pixels, thus the latency or amount of transmission time is doubled from 109 milliseconds to 252 milliseconds. Visually it can be seen that the larger resolutions have a wider range in transmission time and this disparity can be attributed to the fact that the longer the transmission is occurring, there is a higher possibility of errors or interferences, thus an increase in transmission time. This can be further deduced by the fact that the maximum for the 320x240 resolution is 504 ms with outliers reaching thousands of milliseconds at a higher rate. As for the time for the smallest resolution of 96x96 pixels, similarly, as expected, a small decrease in transmission time is seen since the number of bytes is only minimal. With these results, it can be interpolated that the amount of time to transmit a 64x64 pixel image would take 41 milliseconds, which is a considerable decrease. This is calculated by using a similar ratio of transmission time and number of pixels as the 96x96 resolution since it is the closest in size.

As stated earlier, a second experiment was conducted choosing resolutions that would be captured and transmitted if no pre-processing was done and only the resolutions supported by the camera driver were an option. This in turn sets the resolutions to 96x96, 176x144, and 400x296 pixels, which gives the necessary dimensions and information to resize the images to the tested resolutions of 64x64, 126x126, and 256x256. Below are the results of the second experiment.

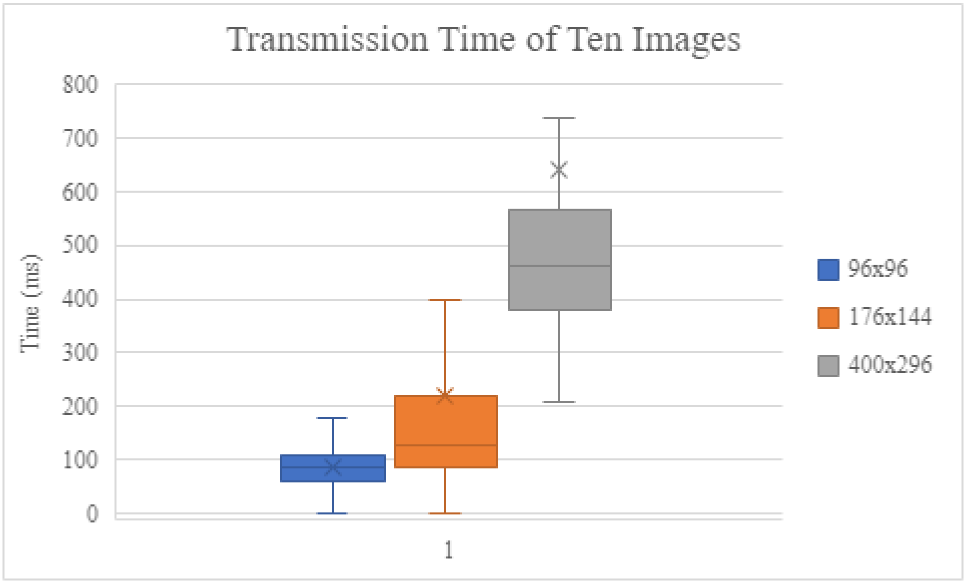

Figure 19. Transmission Time of Ten Images for Second Set of Resolutions

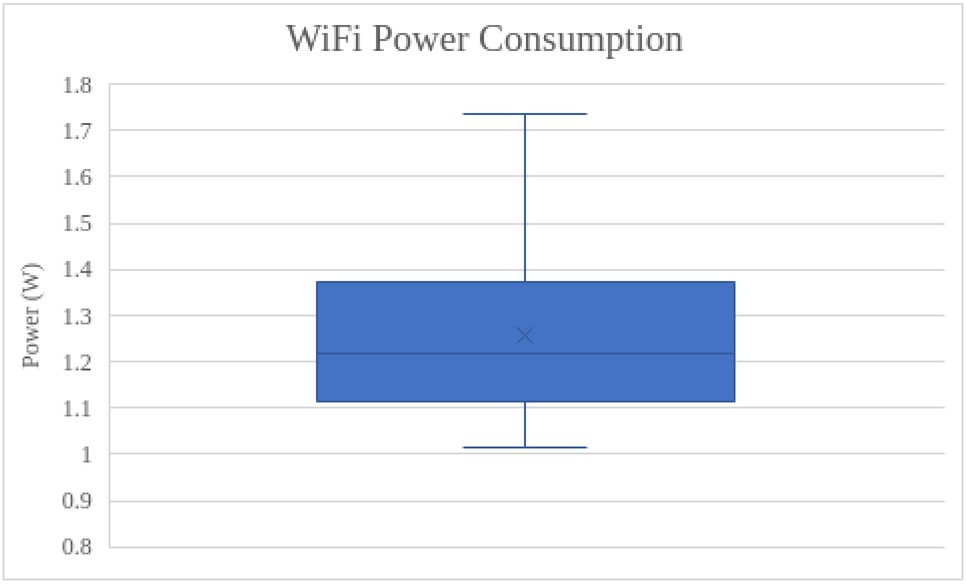

Figure 20. Wi-Fi Power Consumption During Transmission of Images

From the above results it is apparent the correlation of the size of the image and the time of transmission. The reduction in time from the largest resolution of 400x296 pixels, and the smallest of 96x96 pixels, is about 81% and assuming the transmission time for a 64x64 image is roughly 41 milliseconds, there is a reduction of 91%.

Power consumption was also sampled and displayed in a box and whisker plot. There was considerable variability in values but on average the consumption was around 1.2 watts. Resolution of the image does not affect this value.

6.2.3: FINAL SYSTEM

Where seconds matter in emergency situations there is also the need for optimal overall system energy consumption. Monitoring and data collection needs to be constant throughout the waking hours of the user, and the possibility of interruption occurring due to loss of power needs to be reduced. Thus, the system parameters chosen for the wearable device and the companion robots classification system are set to the necessary values that result in an accuracy of 94.74%. This accuracy is achieved by having a model with a set fully connected layer, following the

ResNet-152 convolutional neural network, of size 320 and 240, with the flattening layer outputting a dimension of 256. Also, the LSTM of three hidden layers is set to the node size of 256 and the final fully connected layer to 126 nodes. Along with this model, the image resolution captured by the wearable device is set to 96x96 pixels. This system gives the optimal performance for the application at hand. A 94.74% accuracy is sufficient since the system can rely on the option of asking the user if they are ok and do not need medical attention. This system sees a reduction in transmission time or energy consumption up to 31%, a 2% reduction in the ResNet-152 parameters, and a sizable 24% reduction in the LSTM parameters.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

7.1: CONCLUSION

In conclusion, the collaborative system is able to minimize its power consumption by reducing the image size from the original VGA resolution of 800x600 to 64x64 and still retain a considerable amount of accuracy, seeing a 0.41% increase in the overall score. This thesis lays a foundation for further research into the collaborative system in both areas, the wearable, and the robot companion. The goal is to optimize for efficiency in speed and power consumption to prolong the life of the wearable and give the best response time necessary for emergency situations which was achieved.

7.2: FUTURE WORK

Over the course of the thesis, several suggestions were mentioned that could further improve the overall collaborative system to classify the user activities better and efficiently, either for fall detection in case of injury or less emergent situations like dietary monitoring or to analyze the routine of the elderly adult. The easiest improvement and the most important in validating the system is by collecting more image data. Most neural networks have datasets in the thousands or even in the millions, like the ILSVRC dataset, for training purposes.

A point-of-view dataset for activity classification is a niche subject but could help eventually for the purpose of this system and other research opportunities. It is important that different environments with different lighting and objects are used so the system is more robust and less susceptible to changes. Currently only an individual location is used and the change in lighting is only due to sunlight exposure for the different days of data collection.

Another component that could see improvements in the system is the change in parameters such as the number of images taken and used to train and the number of ResNet layers. Ten is an arbitrary number of images and could be an ineffective number, either too many or too few. This could reduce the transmission latency or improve the overall classification accuracy. Similarly, for the ResNet layers, the 152-layer architecture was not tested against any other architectures and could be reduced for a higher accuracy or argued that a reduction in parameters and classification time is more important for the system.

REFERENCES

1. David, Uchechukwu Gabriel, "A Cloud Infrastructure for Large Scale Health Monitoring in Older Adult Care Facilities" (2021). Masters Theses. 1148.
2. Morris A. E-literacy and the grey digital divide: A review with recommendations. J Inf Literacy. 2007;1(3):13–28. doi: 10.11645/1.3.14.
3. World Health Organization. (2021, October 4). Ageing and health. World Health Organization. Retrieved July 25, 2022, from https://www.who.int/news-room/fact-sheets/detail/ageing-and-health
4. G. Schiboni, F. Wasner and O. Amft, "A Privacy-Preserving Wearable Camera Setup for Dietary Event Spotting in Free-Living," 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2018, pp. 872-877, doi: 10.1109/PERCOMW.2018.8480222.
5. Sun, M., Burke, L. E., Mao, Z. H., Chen, Y., Chen, H. C., Bai, Y., Li, Y., Li, C., & Jia, W. (2014). eButton: A Wearable Computer for Health Monitoring and Personal Assistance. Proceedings. Design Automation Conference, 2014, 1–6. https://doi.org/10.1145/2593069.2596678
6. K. Zhan, F. Ramos and S. Faux, "Activity recognition from a wearable camera," 2012 12th International Conference on Control Automation Robotics & Vision (ICARCV), 2012, pp. 365-370, doi: 10.1109/ICARCV.2012.6485186
7. N. Zerrouki, F. Harrou, A. Houacine, and Y. Sun, “Fall detection using supervised machine learning algorithms: a comparative study,” in Proceedings of the 8th International Conference on Modeling, Identification and Control (ICMIC), pp. 665–670, Algiers, Algeria, November 2016.
8. A. Shojaei-Hashemi, P. Nasiopoulos, J. J. Little, and M. T. Pourazad, “Video-based Human TelegramFall Detection in Smart Homes Using Deep Learning,” Proceedings - IEEE International Symposium on Circuits and Systems, vol. 2018-May, pp. 0–4, 2018.
9. M. Casares, K. Ozcan, A. Almagambetov and S. Velipasalar, "Automatic fall detection by a wearable embedded smart camera," 2012 Sixth International Conference on Distributed Smart Cameras (ICDSC), 2012, pp. 1-6.
10. L. Martinez-VillasenoTelegramr, H. Ponce, and K. Perez-Daniel, “Deep learning for multimodal fall detection,” Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics, vol. 2019-October, pp. 3422–3429, 2019.
11. K. Ozcan and S. Velipasalar, “Wearable Camera- and Accelerometer-Based Fall Detection on Portable Devices,” IEEE Embedded Systems Letters, vol. 8, no. 1, pp. 6–9, 2016.
12. F. Erivaldo Fernandes, H. M. Do, K. Muniraju, W. Sheng, and A. J. Bishop, “Cognitive orientation assessment for older adults using social robots,” in 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2017, pp. 196–201.
13. HHTseng (2020) video classification (Version 0.0) [Source Code]. https://github.com/HHTseng/video-classification.

14. Rémi Mégret, Vladislavs Dovgalecs, Hazem Wannous, Svebor Karaman, Jenny Benois-Pineau, Elie El Khoury, Julien Pinquier, Philippe Joly, Régine André-Obrecht, Yann Gaëstel, and Jean-François Dartigues. 2010. The IMMED project: wearable video monitoring of people with age dementia. In Proceedings of the 18th ACM international conference on Multimedia (MM '10). Association for Computing Machinery, New York, NY, USA, 1299–1302. https://doi.org/10.1145/1873951.1874206
15. P. Mazurek and R. Z. Morawski, "Application of naïve Bayes classifier in fall detection systems based on infrared depth sensors," 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015, pp. 717-722, doi: 10.1109/IDAACS.2015.7341397.
16. Liu, Chien-Liang, Chia-Hoang Lee and Ping-Min Lin. "A fall detection system using k-nearest neighbor classifier." Expert Syst. Appl. 37 (2010): 7174-7181.
17. Charfi, Imen, Johel Mitéran, Julien Dubois, Mohamed Atri and Rached Tourki. "Definition and Performance Evaluation of a Robust SVM Based Fall Detection Solution." 2012 Eighth International Conference on Signal Image Technology and Internet Based Systems (2012): 218-224.
18. Juang, Chia-Feng and Chia-Ming Chang. "Human Body Posture Classification by a Neural Fuzzy Network and Home Care System Application." IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 37 (2007): 984-994.
19. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
20. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
21. O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.
22. Barry, R. (2020, September 30). Decrease Ram footprint and accelerate execution with FreeRTOS notifications. FreeRTOS. Retrieved November 14, 2022, from https://www.freertos.org/2020/09/decrease-ram-footprint-and-accelerate-execution-with-freertos-notifications.html

VITA

RICARDO HERNANDEZ

Candidate for the Degree of

Master of Science

Thesis: POWER-EFFICIENT COLLABORATIVE FALL DETECTION USING A WEARABLE DEVICE AND A ROBOT

Major Field: ELECTRICAL ENGINEERING

Biographical:

Education: Bachelor’s of Science in Computer Engineering

Completed the requirements for the Master of Science in your major at Oklahoma State University, Stillwater, Oklahoma in December, 2022.

Completed the requirements for the Bachelor of Science in your major at Oklahoma State University, Stillwater, Oklahoma in 2019.

Experience:

Professional Memberships: