GESTURE RECOGNITION THROUGH WI-FI SENSING IN SMART HOMES

A CULMINATING EXPERIENCE REPORT
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF SAN FRANCISCO STATE UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER

KRUNAL SHAH

JULY 2019

Abstract

With the fast growing of computer technology, smart home has become a necessary trend for future life. In smart homes, people trend to have control over all their devices and appliances. Thus, human gesture recognition has become critically important in designing smart homes and management services. As for senior people, it can be useful if a person is suffering from social isolation or having other health problems. Gesture recognition system can detect any changes in their activity and call for emergency if needed. It can also be used for monitoring on-going actives of the user, and also for security purpose. In this project, we propose a novel alternative sensing method for smart homes, which utilizes the prevalent Wi-Fi devices and networks at home to detect and recognize human gestures. Specifically, we build a machine learning model with autoencoder and softmax regression model to examine and extract the unique patterns exhibited in the Channel State Information available in the received Wi-Fi signals at wireless devices and use it to sense and identify human gestures in indoor environment. Experimental results show that even though Wi-Fi based gesture recognition is a relative new area of research, there were some limitations but we achieved a very good accuracy of 93.8%.

Acknowledgements

First and foremost, I would like to express my sincere appreciation to my advisor, Prof. Hao Yue, for his invaluable guidance and strong support during my master's Thesis Project. I have learned so much from him about being a serious thinker and researcher.

I would like to thank all my professors at San Francisco State University, their efforts were invaluable towards my academic progress and becoming skillful. My gratitude
to all my fellow classmate and friends who helped me.

Finally, I owe a special debt of gratitude to my parents. Thank you for supporting me all the way to today, even in my most difficult time. I love you!

Contents

List of Tables

List of Figures

1 Introduction

With the fast growth of smart appliances and smart devices at home, smart homes have received increased attentions and have been translating from a vision into reality. In smart homes, we need to track and recognize the activities and gestures of residents to intelligently control home appliances and achieve comfort, convenience, security, and energy efficiency. The main applications of the gesture recognition can be used for the following purposes: Firstly, it can be used for monitoring of inhouse exercises, human respiration and heart rates while sleeping. Secondly it can be used to preserve energy by using wi-fi based gesture recognition for switching on and off devices and automatically shifting the deice to deep sleep when no human movements is detected.

Traditional approaches on gesture recognition are often device-based approaches which requires installing dedicated devices or sensor. For example, in sensor-based approach the user needs to install sensors or radars for detecting human gestures. While in camera based, the user needs to perform gesture in the line-of-sight of camera. And in wearable-device based approach the gesture is detected by device like smart watches, smart gloves which can be wore in day to day life. However, the drawbacks of these device-based approaches are: Firstly the user needs to wear a wearable-based device which can be uncomfortable and an overhead. Secondly for camera-based approach, the camera needs to be in line-of-sight of human gestures to detect it, that may require to install multiple camera that will eventually increase the cost as well as privacy issues.

In this project, we develop a new system for recognizing human gestures with Wi-Fi signals. We extracted Channel State Information (CSI) from the Wi-Fi signal. Then, we utilize machine learning approach with autoencoder and softmax regression to train the algorithm and recognize human gestures.

With the fast growth of smart appliances and smart devices to home, Smart Homes have received increased attention and have been translating from a vision into reality. Recently human activity monitoring technology have become critically important with its wide variety of uses like security, sports and fitness activities, healthcare of elderly people, etc.

The US health care system faces daunting challenges. With the improvements in health care in the last few decades, residents of industrialized countries are now living longer, but with multiple,

often complex, health conditions. Finally, recent health care reform efforts may add 32 million newly insured patients to the health care system in the next few years.
Also there has been a lot of research for finding new methods for authenticating. Few years ago, we used to have passcodes and locks for authenticating the users. Recently it is being replaced by figure print recognition or Iris/face recognition. But still these methods are less effective and very much expensive, so a lot of people are not able to use it.

Additionally, traditional human monitoring techniques require to use sensors or some dedicated device for identifying human movements and are often expensive. Camera based human gesture recognition are less faulty and highly robust, but they require the activity to be performed in line-of-sight of the camera. So, in many places we may need to install multiple devices which can eventually increase the cost, also monitoring human gestures through camera are risk oriented for personal security. No one would like to install a camera in their personal spaces.

However, part of the solution may lie in how and to what extent we take advantage of recent advances in information technology and related fields. Wi-Fi can be used to detect human movements and then ultimately identify human gestures. According to recent studies, we can identify human gesture by studying Received Signal Strength Information (RSSI) and Channel State Information (CSI) from physical level of the Wi-Fi devices.

Wi-Fi based gesture recognition would eliminate all the disadvantages laid by sensors and camera. Moreover, all majority of indoor places are equipped with Wi-Fi nowadays, so there would be no need to install any dedicated device and thus it would be a great cost cutter. There would be no need to install any sensors or wear any wearable devices, as Wi-Fi is easily available at almost every place, it also doesn't need to charge and would never run out of battery. Also, the privacy concerns with camera-based techniques are eliminated by using the Wi-Fi for detecting human gestures.

Thus, we have developed a Wi-Fi based human gesture recognition system that does not require any additional devices and can be easily implemented in Smart Homes.

The rest of the report is organized as follow. In Section 2, we review the related work and analyze the limitations of existing methods. In Section 3, we elaborate the design of the gesture recognition system. We next show the system setup and experimental results in Section 4. Finally, we conclude our work in Section 5.

2 Related Work

Due to its attractive potential application in smart homes, human monitoring and gesture recognition have witnessed great efforts in exploring its potential. In this section, we briefly review the development of this technique and compare the existing solutions. There are mainly two types of approaches for detecting human movements: device-based methods and device-free methods.

Device-based approaches use dedicated devices that need to be installed like sensors, wearable devices, smart watches, radars, etc., to monitor human movements [7, 8]. Device-based approaches are often costly as they require installing extra devices. Also, the user needs to constantly wear a device to his body that can be uncomfortable and an overhead.

Device-free approaches would help to overcome such limitations by using Wi-Fi signals for human gesture recognition. Nakatsuke et al. [9] used two main features obtained from wireless sensors called Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LSI) to perform crowd estimation in indoor environment. But the main disadvantage of this method is that it requires extensive deployment of sensor nodes which increases the cost of this approach highly with laborious training efforts. There has also been another approach-based camera-based technique [10] which performs crowd density estimation by studying video images and then relate to foreground pixel count to the number of people. However, for this approach high deployment cost and privacy concerns limits it's wide spread use.

To overcome all these limitations, in this project, we use Channel State Information (CSI) in Wi-Fi signals to recognize human gestures. Nowadays, CSI has been used in various applications including user-authentication [15], indoor localization, activity recognition [13]. Wu et al. [14] have used CSI to assist indoor localization by using Line of Sight (LOS) in indoor environment. They have studied correlation between CSI signals and estimated Angle of Arrival (AoA) based on phase difference between different antennas. WiFall [16] uses CSI to detect fall movements for individual person. This kind of detection is really useful specially for the elderly homes. They have used one-class Support Vector Machine (SVM) to detect fall movement from every other movement. On the other hand, our project aims to provide fine-grained smart human monitoring system that require no additional use of external device and which is very easy to use. It is a software-only solution that works with Wi-Fi beacon signals, which is easily deployable and low-cost.

3 Design and Implementation

3.1 System Design and Overview

Figure 3.1 shows the overview of our human gesture recognition. There are four main steps:

1. Data Collection
2. Data Pre-processing
3. Deep Learning Classifier
4. Recognizing Gesture

The first step consists of CSI collection through a set of Wi-Fi enabled devices which are setup as a transmitter and a receiver. After the data is collected, it is pre-processed before feeding it into a deep learning classifier. In the pre-processing step, we first remove the unwanted environmental noise by preforming denoising and subcarrier selection. After the data is cleaned, it is segmented into two parts: Training data and Testing data. Once all the pre-processing of data is finished, the training data is feed into the deep learning classifier consisting of an Autoencoder a Softmax Layer. The output of the classifier is compared against the testing data and we get the accuracy.

Once the system is trained, we use the deep learning classifier for real-time gesture recognition system. A Graphical User Interface (GUI) is also created to display the signal and the gesture in real-time.

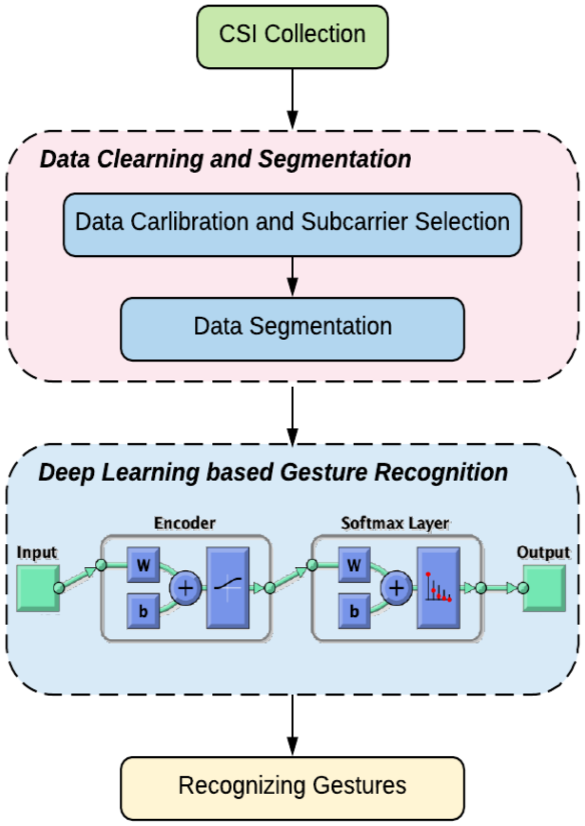

CSI Collection
Data Clearning and Segmentation
Data Carlibration and Subcarrier Selection
Data Segmentation
Deep Learning based Gesture Recognition
Encoder
Softmax Layer
Input
Output
W
b
W
b
Recognizing Gestures

Figure 3. 1: System Overview

3.2 Data Collection

3.2.1 CSI Tools

We have used CSI tools to collect data for our gesture recognition system. CSI describes how a signal propagates from the transmitter to the receiver and represents the combination effect of, for example, scattering, fading, and power decay with distance. The quantitative analysis of signal propagation behavior within a Wi-Fi covered area can identify different types of disturbances. The CSI tool is built on the Intel Wireless Network Interface Card (NIC).

As Wi-Fi signals propagate in physical space, the signals reach receivers (Wi-Fi devices) through various routes, a concept known as multipath. The received signal is composed of signals arriving over many different paths, all which can be affected by environmental factors. The environmental factors therefore combine with scattering, fading, and power decay over distances, and these environmental effects on the signals can be manifested in the CSI. For example, while CSI remains stable in a static home environment, if a person performs activities, Wi-Fi signals scatter in response to the body's movement, thereby causing bistatic Doppler shift at the receiver. The bistatic Doppler frequency depends on the occupant's moving speed**,** multipath propagation of Wi-Fi signal indoors, wi-fi frequency band, and the relative position between the occupant and the Wi-Fi transceivers. The IWL5300 provides 802.11n channel state information in a format that reports the channel matrices for 30 subcarrier groups. Each channel matrix entry is a complex number, with signed 8-bit resolution each for the real and imaginary parts. It specifies the gain and phase of the signal path between a single transmit-receive antenna pair [19].

Our system takes time-series of raw CSI measurement. Each input file contains 30 N_t x N_r matrices, where N_t and N_r are the number of antennas on transmitters and receiver respectively. The data is collected for a single person, performing four different gestures: walking, sitting, standing, and no gesture (empty room). Thus, we have used CSI, which is very good for recording signals, to record human movements through which we can do gesture recognition.

3.3 Data Pre-Processing

The raw CSI data contains high frequency noise, outliers, and artifacts. We use wavelet-based denoising to remove the high frequency noise from the raw CSI data. The filtered data became smooth after the high-frequency noise reduction. Wavelet based denoising is also used to denoise signals and images. Wavelets are capable of localizing features in our data to various different scales which in turn helps in preserving important features of the signals by removing the noise. This approach preserves the high frequency features in the signal/image and therefore does a considerably better job of denoising.

When we input the signal for wavelet based denoising it is passed through two filters: one is low-pass filter and the other is high-pass filter. The results from the low-pass filter is called approximation and the output from high-pass filter is called details. Approximation, demoted as A, contains the low-frequency, high-scale components of the original signal. Details, denoted as D, contains high-frequency, low-scale components of the original signal. This process of decomposition can be carried out for a number of times so that successive approximation is decomposed, and each signal is broken down into lots of lower resolution components. This is called wavelet decomposition tree, shown in the below figure:

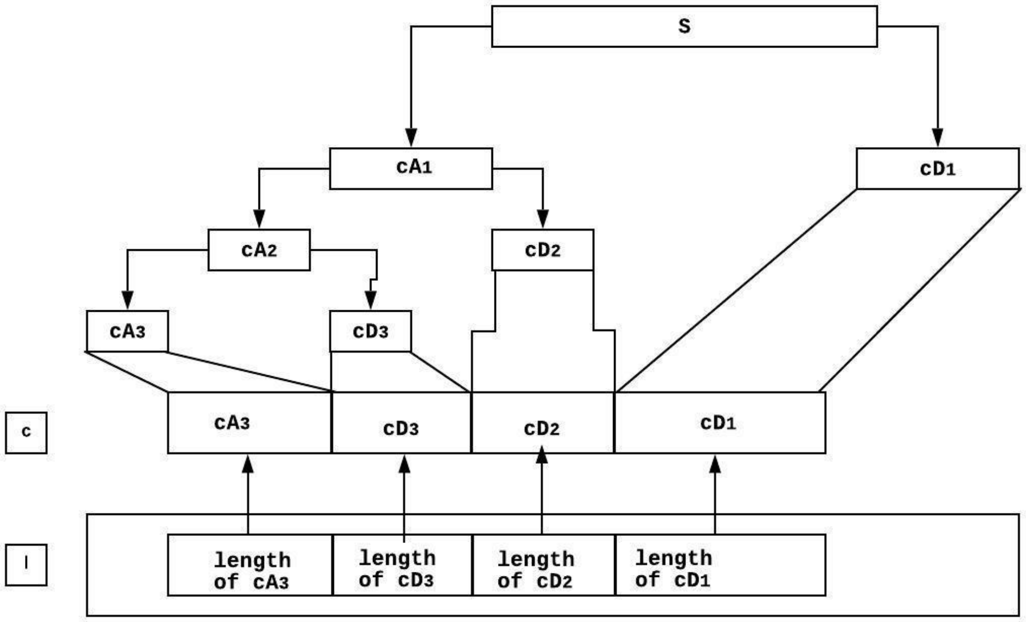


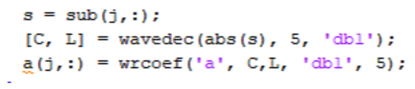
Figure 3. 2: Wavelet packet decomposition tree

In Matlab, c denotes the resulted approximation and details stored in a vector, and l, denotes the length of these components. Once the decomposition is done, we then reconstruct the signal and remove the noise. The original signal is constructed via only approximation components, as results from wavelet-based decomposition will contain the environmental noise.

We have used two Matlab functions wavedec and wrcoef to denoise the signals.

- wavedec – Matlab function used for decomposition
- wrcoef – Matlab function for selective reconstruction

Figure 3.2 shows the Matlab code for using these two functions to perform level-5 denoising.



```
s = sub(j,:);
[C, L] = wavedec(abs(s), 5, 'db1');
a(j,:) = wrcoef('a', C,L, 'db1', 5);
```

Figure 3. 3 : Denoising in Matlab

Here are the images of signals before and after denoising. As we can clearly see that denoising makes the signal smooth.

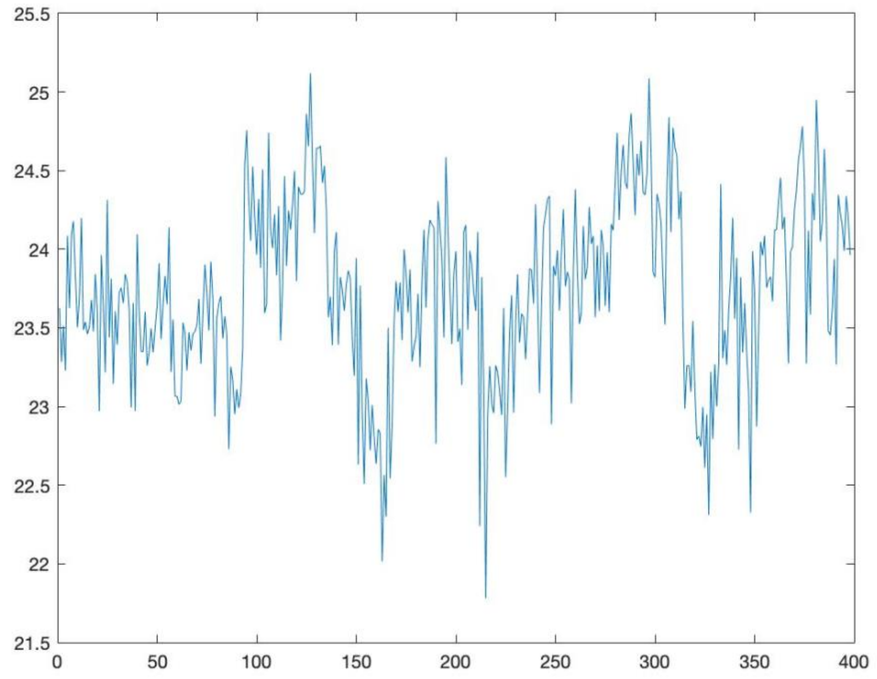


Figure 3. 4: Signal with noise

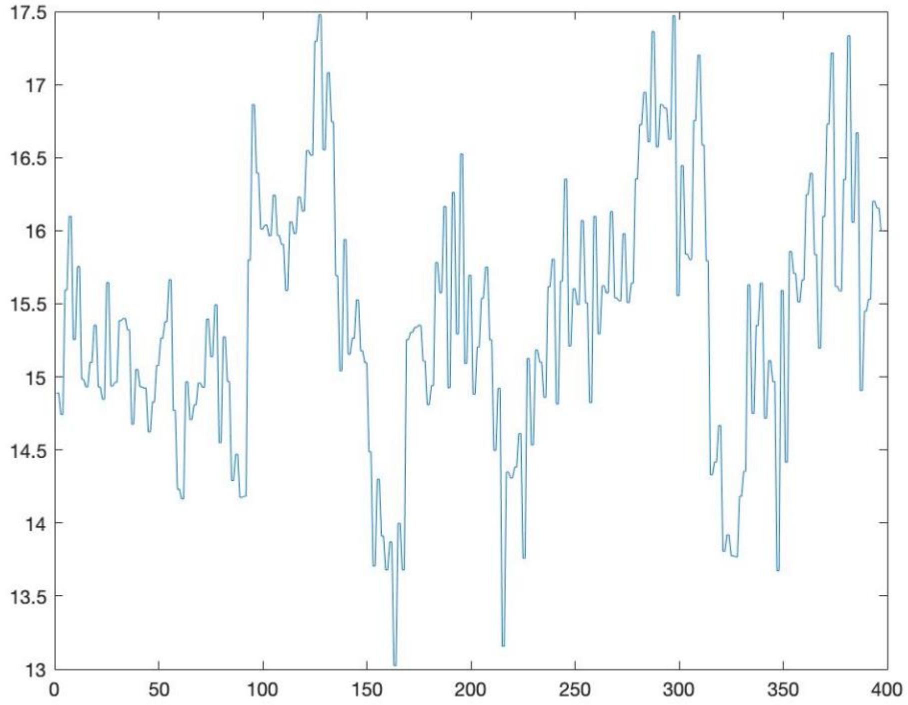


Figure 3. 5: Normal and denoised Signal

Subcarrier selection: We did extensive testing for data collected with single sub-carrier and also with 30 sub-carrier, the latter one got much higher accuracy. So, for our final testing phases we decided to go with all 30 sub-carrier to get the best accuracy.

Data segmentation: After pre-processing the data, it is split into two sets: Training-data and Testing-data. The data is divided in the ratio 8:2, 80% of the data is used for training purpose and the rest of 20% is used for testing purpose. We have also provided labels for these data based on the four classes: walking, standing, sitting, no gesture (empty room).

3.4 Deep Learning based Algorithm for gesture recognition

3.4.1 Machine Learning

Machine Learning is a concept which allows the machine to learn from examples and experience, and that too without being explicitly programmed. So instead of writing the code, data is feed to the generic algorithm, and the algorithm/machine builds the logic based on the given data.

Machine Learning is an idea in which we train the machine to learn from experience and examples. As we humans are evolving constantly by learning from our past experiences, similarly we can train a machine to learn from given examples and give the outputs based on its learning. Machine Learning algorithm is trained using a training data set to create a model. When new input data is introduced to the ML algorithm, it makes a prediction on the basis of the model. The prediction is evaluated for accuracy and if the accuracy is acceptable, the Machine Learning algorithm is deployed. If the accuracy is not acceptable, the Machine Learning algorithm is trained again and again with new training data set.

This is just a very high-level example as there are many factors and other steps involved. There are three main types of Machine Learning Algorithm:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

Supervised Learning is the one, where you can consider the learning is guided by a teacher. We have a dataset which acts as a teacher and its role is to train the model or the machine. Once the model gets trained it can start making a prediction or decision when new data is given to it.
E.g. Logistic Regression, Naive-Bayes, SVM, KNN, Artificial Neural Network

The Unsupervised Learning model learns through observation and finds structures in the data. Once the model is given a dataset, it automatically finds patterns and relationships in the dataset by creating clusters in it. What it cannot do is add labels to the cluster, like it cannot say this a group of apples or mangoes, but it will separate all the apples from mangoes. Suppose we presented images of apples, bananas and mangoes to the model, so what it does, based on some patterns and relationships it creates clusters and divides the dataset into those clusters. Now if a new data is fed to the model, it adds it to one of the created clusters. E.g. K-means, SVD, PCA

Reinforcement Learning is the ability of an agent to interact with the environment and find out what is the best outcome. It follows the concept of hit and trial method. The agent is rewarded or penalized with a point for a correct or a wrong answer, and on the basis of the positive reward points gained the model trains itself. And again, once trained it gets ready to predict the new data presented to it.

3.4.2 Neural Networks

Neural networks are Machine Learning algorithms that are modeled similarly to the way human brains works. An example for a neural network algorithm can be of image recognition, an algorithm that can recognize if an image has dog or doesn’t have a dog [20].
These are the five basic building blocks of a neural network

1. Neurons
2. Inputs
3. Outputs (Activation Function)
4. Weights
5. Biases

Neurons are the decision makers. Each neuron has one or more inputs and a single output called an activation function. This output can be used as an input to one or more neurons or as an output for the network as a whole. Some inputs are more important than others and so are weighted accordingly. Neurons themselves will "fire" or change their outputs based on these weighted inputs. How quickly they fire depends on their bias. Here's a simple diagram covering these five elements.

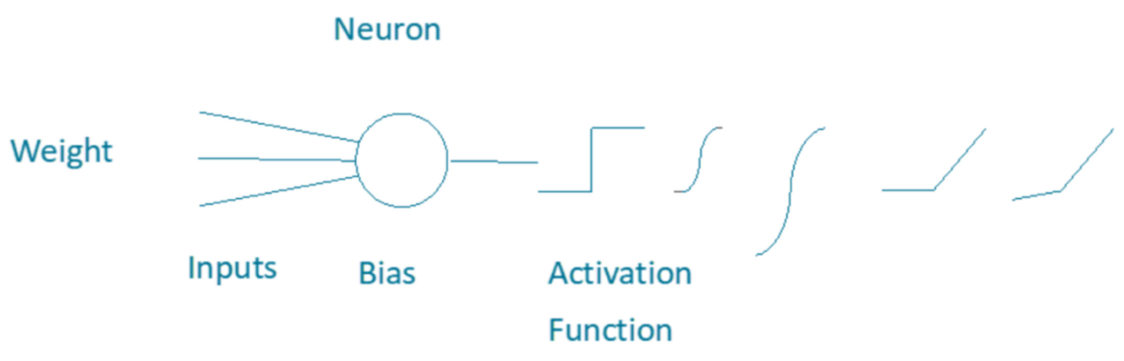


Figure 3. 6: Neural Networks Building Blocks [3]

There are no weights and bias in this diagram, but you can think of them as floating-point numbers, typically in the range of 0-1. The output or activation function of a neuron doesn't have to be a simple on/off (though that is the first option) but can take different shapes. In some cases, such as the third and fifth examples above, the output value can go lower than zero. And that's it!

Here's a network that is used to recognize handwritten digits.

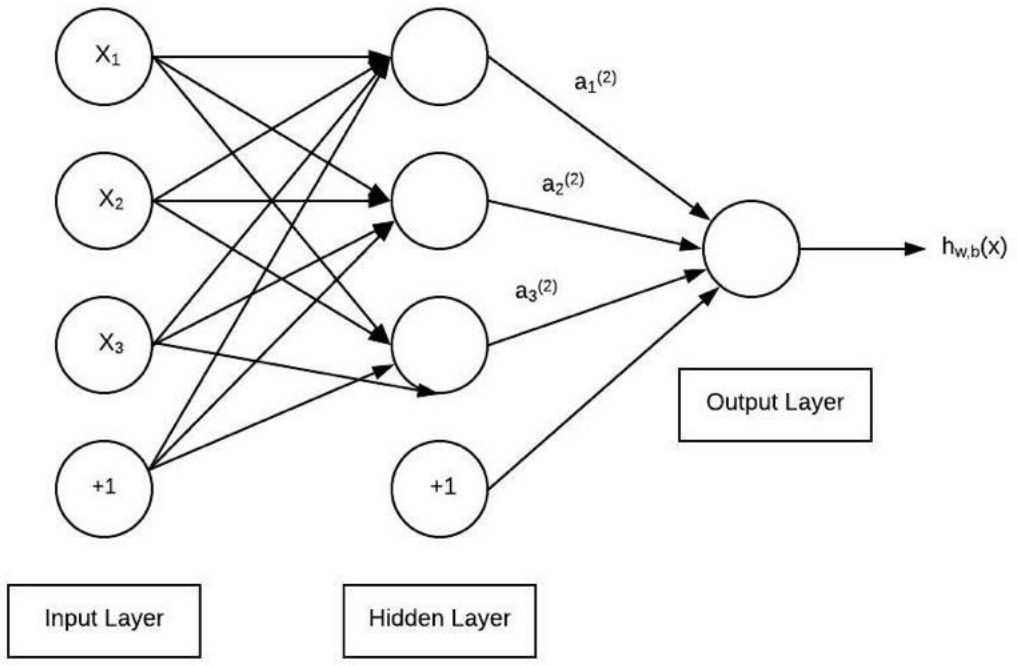


Figure 3. 7: A Neural Network to recognize handwritten digits [3]

There are many other approaches to neural networks that have different strengths and weaknesses or are used to solve different types of problems. But concepts here still apply. Neural networks are all built on the same basic elements: neurons (with bias), inputs (with weights), and outputs (activation functions with specific profiles). These elements are used to construct different or specialized layers and elements (like the LSTM unit above). All of these things are combined with feedback loops and other connections to form a network.

3.4.3 Autoencoder

Artificial Intelligence encircles a wide range of technologies and techniques that enable computer systems to solve problems like Data Compression which is used in computer vision, computer networks, computer architecture, and many other fields. Autoencoders are *unsupervised neural networks* that use machine learning to do this compression for us.

An autoencoder neural network is an Unsupervised Machine learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. Autoencoders are used to reduce the size of our inputs into a smaller representation. If anyone needs the original data, they can reconstruct it from the compressed data [6].

An autoencoder can learn non-linear transformations with a non-linear activation function and multiple layer. It doesn't have to learn dense layers. It can use convolutional layers to learn which is better for video, image and series data. An autoencoder provides a representation of each layer as the output. It can make use of pre-trained layers from another model to apply transfer learning to enhance the encoder/decoder.

An Autoencoder consists of three layers

1. Encoder
2. Code
3. Decoder

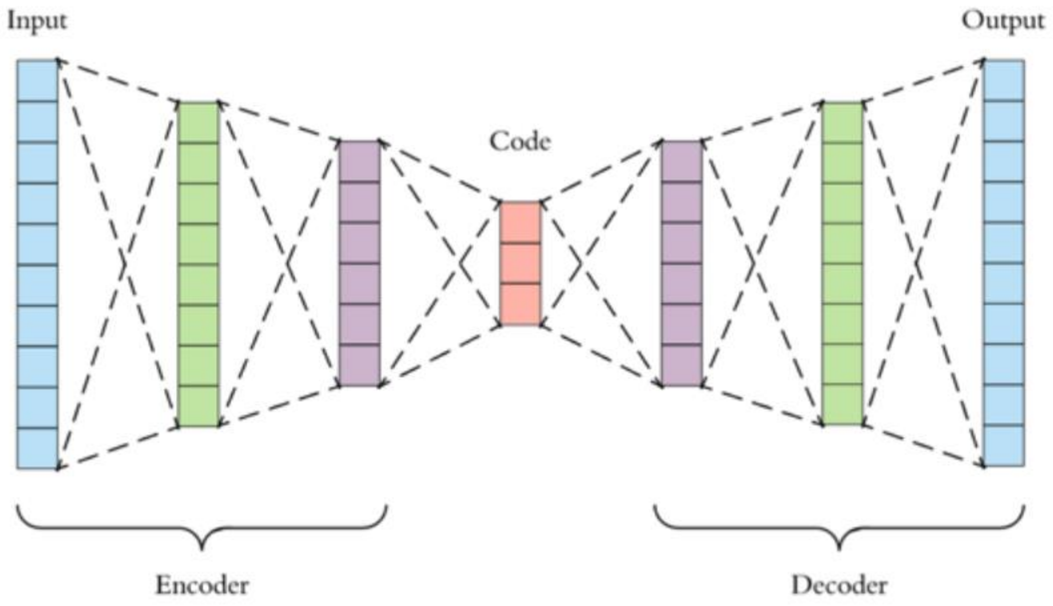


Figure 3. 8: Architecture of Autoencoder [3]

- **Encoder**: This part of the network compresses the input into a latent space representation. The encoder layer encodes the input image as a compressed representation in a reduced dimension. The compressed image is the distorted version of the original image.
- **Code**: This part of the network represents the compressed input which is fed to the decoder
- **Decoder**: This layer decodes the encoded image back to the original dimension. The decoded image is a lossy reconstruction of the original image and it is reconstructed from the latent space representation.

The layer between the encoder and decoder, i.e. the code is also known as Bottleneck. This is a well-designed approach to decide which aspects of observed data are relevant information and what aspects can be discarded. It does this by balancing two criteria:

1. Compactness of representation measured as the compressibility.
2. It retains some behaviorally relevant variables from the input.

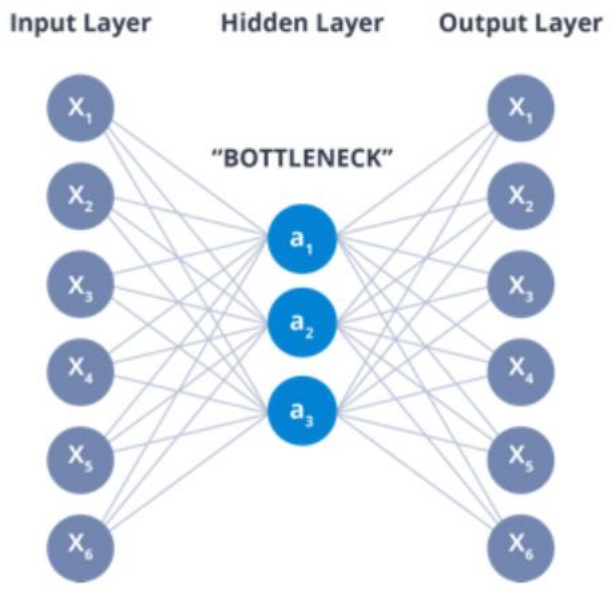


Figure 3. 9: Bottleneck in Autoencoder [3]

Properties of Autoencoder:

- Data-specific: Autoencoder are only able to compress data similar to what they have been trained on
- Lossy: The decompressed outputs will be degraded compared to the original inputs
- Learned automatically from examples: It is easy to train specialized instances of the algorithm that will perform well on a specific type of input.

3.4.4 Softmax Classifier

Softmax Classifier is implemented through a neural network layer just before the output layer. The Softmax layer must have the same number of nodes as the output layer [21].

Softmax extends the idea into a multi-class world. That is, Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise would.

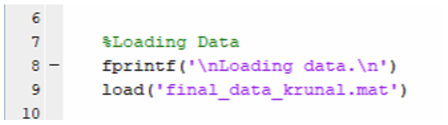
Softmax Classifier, also known as multi class logistic regression is a generalized version of logistic regression. It is used for multi-class classification. In logistic regression, the assumption is that the labels are binary i.e. 0 or 1. While, softmax regression handles $Y \in \{1,2 \ldots k\}$.

Hypothesis function for softmax regression:

$$h_\theta(x) = 1/\sum_{j=1}^{k}(1 + e^{-\theta^T}x)\ [e(-\theta^{(1)T}x);\ e(-\theta^{(2)T}x); \dots\ e(-\theta^{(k)T}x);\ ]$$

3.4.5 Matlab Code

In the first step of our algorithm we feed our pre-processed data and feed it into our Machine Learning algorithm. The preprocessed data can be stored in a ".mat" data file. ".mat" is a file format for data files in MATLAB. The final data file contains 4 variables: training_data, training_labels, testing_data, testing_labels. The code snippet for loading the data file is shown in the below figure:



```
%Loading Data
fprintf('\nLoading data.\n')
load('final_data_krunal.mat')
```

Figure 3. 10: Loading data

Once the data is loaded in the second step, we begin to train a sparse autoencoder by feeding it with the training data set [17]. The training data provided to an Autoencoder is in unsupervised fashion i.e. without any labels. The Autoencoder performs dimensionality reduction for given training data and its features are extracted from its output. These dimensionality reductions are done because the size of the hidden layer is usually much smaller than input layer. So, the autoencoder tries to replicate the input by extracting only important features from the training data. The function and parameters used for training an autoencoder are:

- trainAutoencoder - This function trains a sparse autoencoder with the given input data and other parameters shown below
- hiddenSize1 - Size of hidden representation of the autoencoder, specified as a positive integer value. This number is the number of neurons in the hidden layer
- MaxEpochs- Maximum number of training epochs or Iterations
- L2WeightRegularization- The coefficient for the L_2 weight regularizer in the cost function (LossFunction)
- SparsityRegularization - Coefficient that controls the impact of the sparsity_regularizer in the cost function
- SparsityProportion - Desired proportion of training examples a neuron reacts to

- ScaleData - Indicator to rescale the input data

Now, we should use the encoder from the trained autoencoder to generate the features.

```
autoencl = trainAutoencoder(training_data,hiddenSize1, ...
    'MaxEpochs',300, ...
    'L2WeightRegularization',0.0001, ...
    'SparsityRegularization',1, ...
    'SparsityProportion',0.15, ...
    'ScaleData', true);

feat1 = encode(autoencl,training_data);
```

Figure 3. 11: Training an autoencoder with training data

These features extracted from Auto-encoder are then feed into a softmax classifier function in a supervised manner. So, Labels are provided to the softmax classifier function so that it can train the softmax layer based on the classes. The MaxEpochs are the maximum number of iterations used for training a softmax function. The following the code snippet for training a softmax layer in MATLAB. We will talk about its parameters in Experimental Setup section.

```
softnet = trainSoftmaxLayer(feat1,training_label,'MaxEpochs',300);
```

Figure 3. 12: Training softmax layer

Then the outputs of the autoencoder and softmax layer are stacked together to form a stacked neural network, which can be used for classification. A stacked neural network or Stackednet looks like these:

```
stackednet = stack(autoencl,softnet);
```

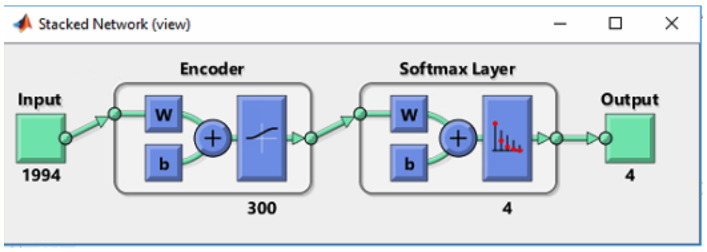
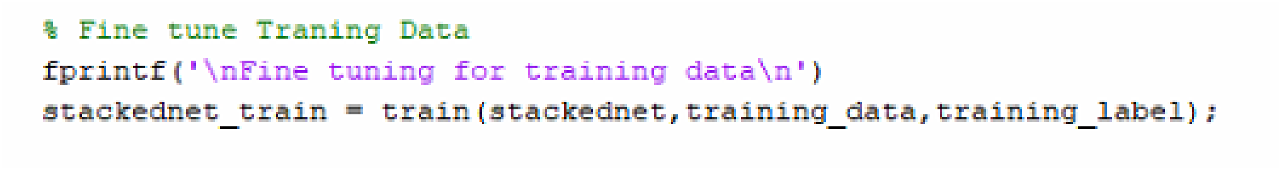


Figure 3. 13: Stacked Neural Network

Once the full network is formed by stacking output of autoencoder and softmax function, we can compute the results on the testing data set.

Fine tuning the stacked neural network: The results for the stacked neural network can be improved by performing backpropagation on the whole multilayer network. This process is often referred to as fine tuning. We can fine tune the network by retraining it on the training data in a supervised fashion. After the full network is formed, we can now compute the results on the testing data.



```
% Fine tune Traning Data
fprintf('\nFine tuning for training data\n')
stackednet_train = train(stackednet,training_data,training_label);
```

Figure 3. 14: Performing Fine Tuning on the Stackednet

You can visualize the results with a confusion matrix. The numbers in the bottom right-hand square of the matrix give the overall accuracy. Through the confusion matrix we are able to know the total number of files that got matched to the labels and which gesture has low accuracy and thus needs to be recorded again.

3.5 Real-Time Monitoring UI

We have developed a Real-Time monitoring system for identifying the gesture performed by user in real-time. In this system we have used our machine learning algorithm along with a MATLAB tools called App Designer to create a simple UI that can display the name of human gesture along with its signal in real-time [18]. The layout of the UI can be shown in the following figure:

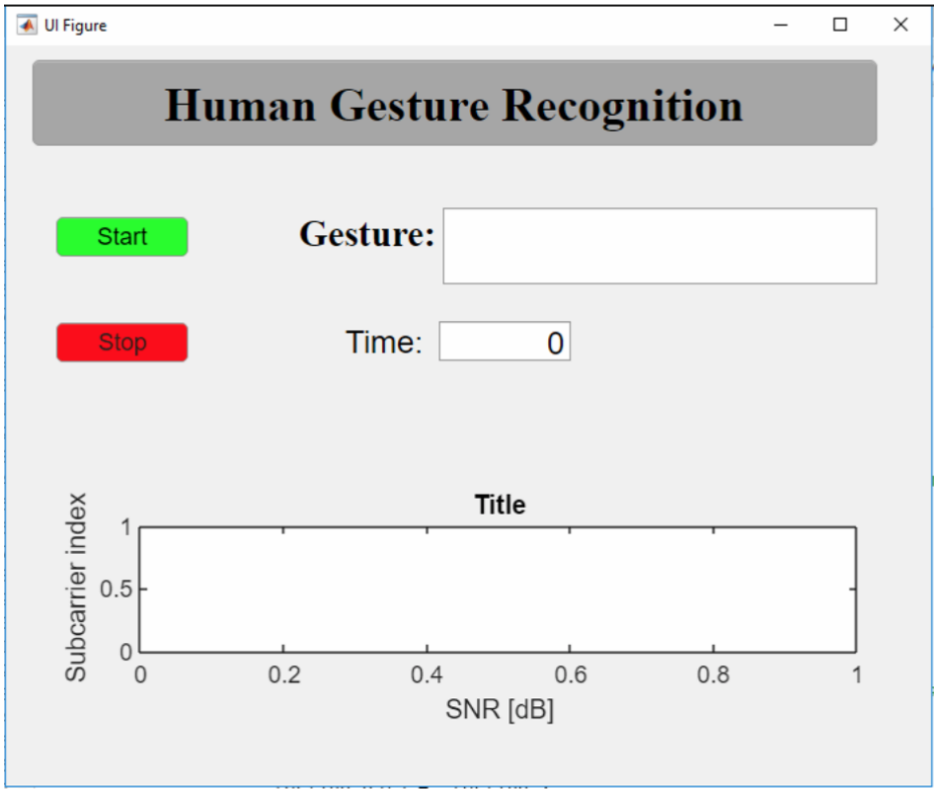
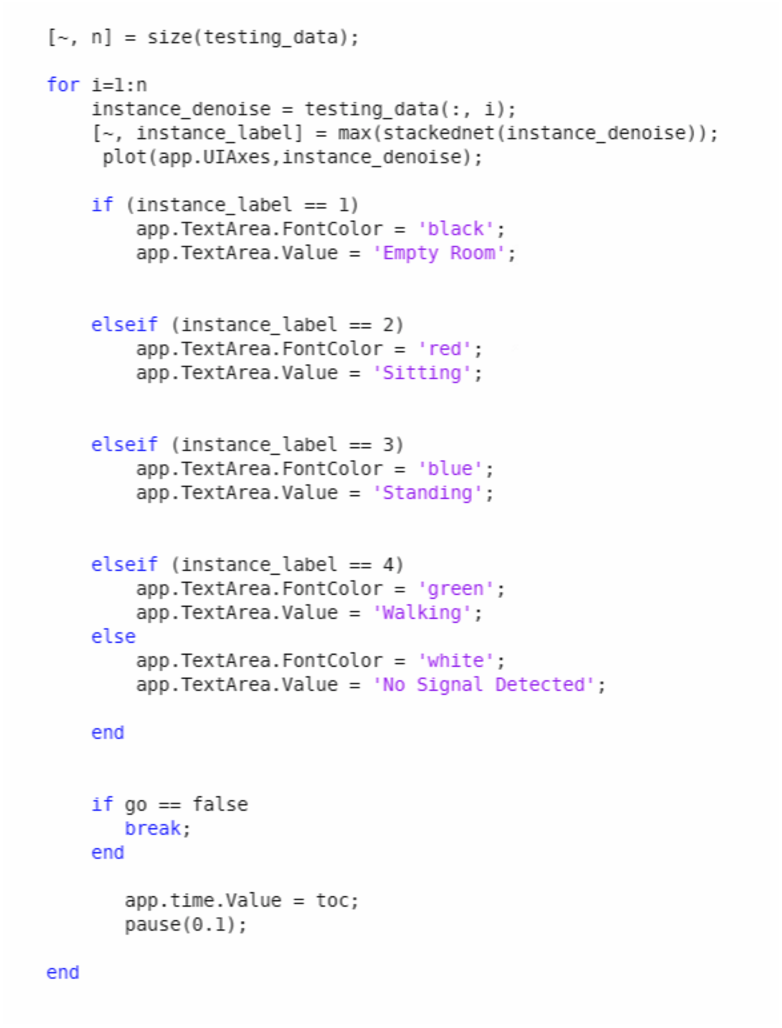


Figure 3. 15: Basic Layout of UI

Initially, we start both the transmitter and the receiver and start to transmit the data, once the data arrives at the receiver, we push the start button in the UI which will start to process the data in real-time. First, we denoise the receiving data, then fragment it into size of 1000 packets each and compare it with the given Stacked Neural Network that was obtained by training our algorithm.

The Stackednet used for the UI is the one in which we got our best accuracy that is of 93% for main four gestures: walking, standing, sitting and empty lab (no gesture). Here is the code snippet that is used for displaying relevant output in our UI



```
[~, n] = size(testing_data);

for i=1:n
    instance_denoise = testing_data(:, i);
    [~, instance_label] = max(stackednet(instance_denoise));
     plot(app.UIAxes,instance_denoise);

    if (instance_label == 1)
        app.TextArea.FontColor = 'black';
        app.TextArea.Value = 'Empty Room';


    elseif (instance_label == 2)
        app.TextArea.FontColor = 'red';
        app.TextArea.Value = 'Sitting';


    elseif (instance_label == 3)
        app.TextArea.FontColor = 'blue';
        app.TextArea.Value = 'Standing';


    elseif (instance_label == 4)
        app.TextArea.FontColor = 'green';
        app.TextArea.Value = 'Walking';
    else
        app.TextArea.FontColor = 'white';
        app.TextArea.Value = 'No Signal Detected';

    end


    if go == false
       break;
    end

       app.time.Value = toc;
       pause(0.1);

end
```

Figure 3. 16: Code for displaying gesture in UI

4 Evaluation

In this section, we evaluate the performance of our system to identify a user's gesture. We have performed the experiments in the university lab.

4.1Experimental Setup

We have used Linux 802.11n CSI Tool for collecting the data [19]. It can be easy downloaded on an Ubuntu with root user access and unencrypted Wi-Fi access point. Once the installation is completed, we can use 2 devices to collect the data. One can be setup as transmitter, which would be used to transmit the signals and the other can be setup as receiver, which would receive the data transmitted by transmitter. The steps to setup the transmitter and receiver are given in below figures.

To imitate the Wi-Fi networks in smart homes, we perform experiments on two laptops that are equipped with 802.11n Wi-Fi NICs (i.e. Intel 5300 NICs). Specifically, we set up two Dell E6430 laptops, one as a transmitter and the other one as a receiver. To measure the CSI, both the laptops run Ubuntu 14.04 operating system and 4.2.0 kernel. The receiver work at Monitor mode, which enables the receivers to capture and decode the 802.11 frames sent by the transmitter. The captured CSI measurements contain the samples from 30 subcarriers, and each subcarrier entry is a complex number with signed 8-bit resolution for both the real and imaginary parts The CSI amplitude is extracted using the links between transmitter's first antenna to receiver's first antenna. The packet transmission rate has been fixed to 1000 packets/seconds.

Activities and environments

All the experiments were carried out in a computer lab at San Francisco State University. The dimensions of the Lab are 25ft x 15ft. The transmitter and the receiver were placed at two ends of the lab as shown in the below figure:

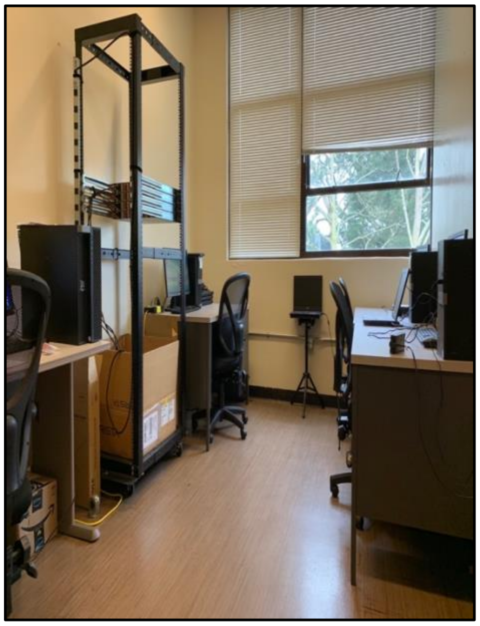


Figure 4. 1: Lab setup

The Lab consists of some PCs, laptops, chairs and table which all remains constant during the whole data collection process. For experiment, one individual performs the gestures of walking, standing and sitting. And another individual records it’s signals in the receiver.

In the below images you can see the person performing the gestures for collecting the data along with the signal recorded for that gesture:

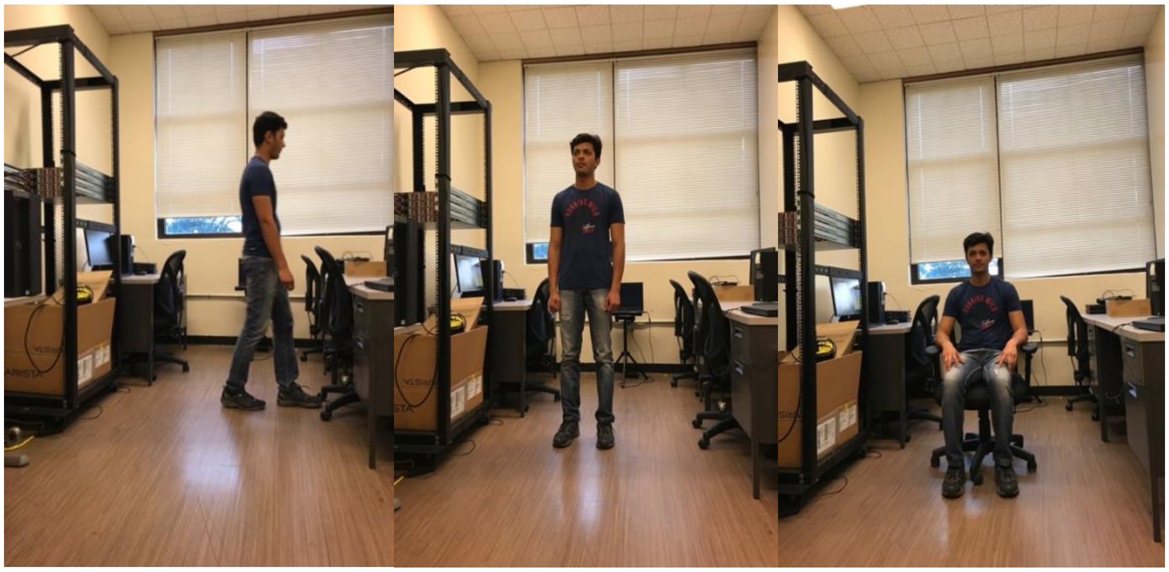


Figure 4. 2: Data collection for Walking, Standing and Sitting

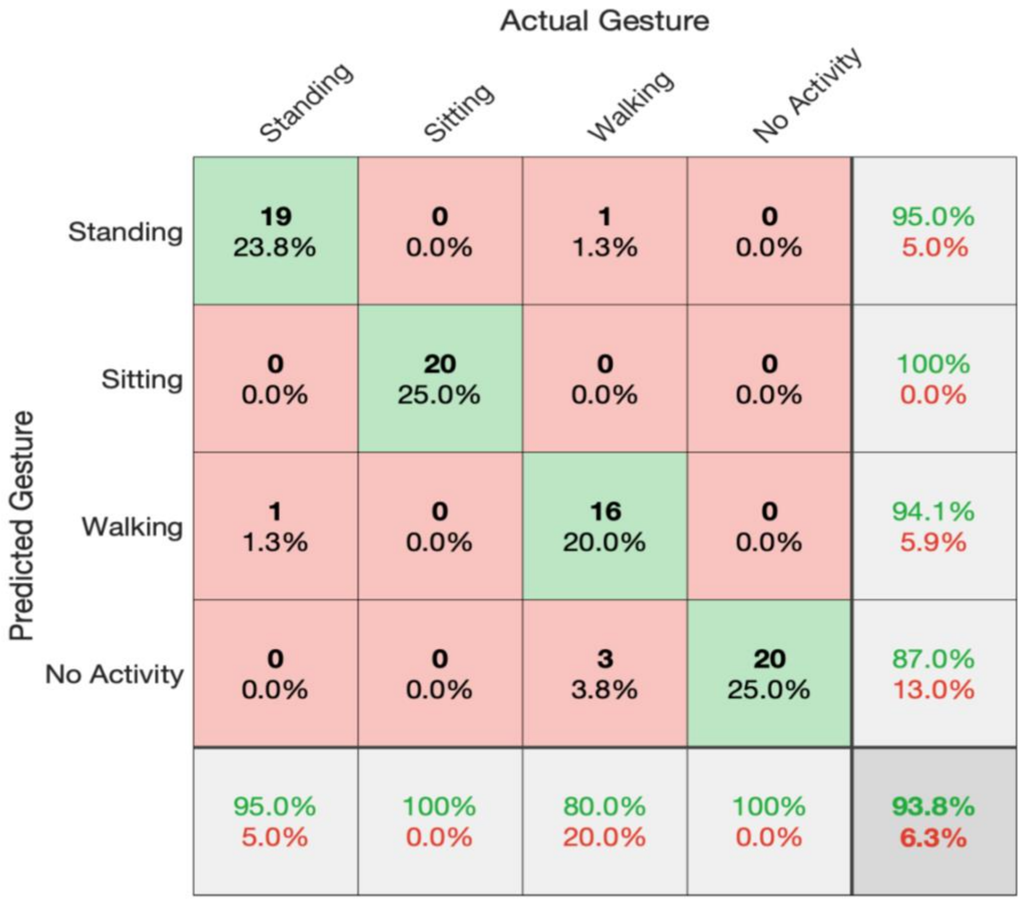
To evaluate the performance of the proposed system, the data is divided into two parts, training data set and testing data set. The total sample of data is divided in the ratio 80% for training and 20% for testing. For this data set we have used a total of 200 samples of each gesture. Then we divided it into two parts 160 samples were used for training and 40 were used for testing. Form the below figure you can see the table of data collected for final testing.

Event Type	Event Description	Total Samples
Walking	Volunteers Walking Freely	200
Standing	Volunteers Standing Upright	200
Sitting	Volunteers Sitting in Chair	200
No Gesture	No Volunteer in room	200

5. 1 Data collection table

4.2 Results

After extensive training and testing of our Deep Learning Algorithm with different data sample the best accuracy that we get is 93.8%. We have shown this result as a confusion matrix in the following figure:



Predicted Gesture \ Actual Gesture	Standing	Sitting	Walking	No Activity	
Standing	**19** 23.8%	**0** 0.0%	**1** 1.3%	**0** 0.0%	95.0% 5.0%
Sitting	**0** 0.0%	**20** 25.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
Walking	**1** 1.3%	**0** 0.0%	**16** 20.0%	**0** 0.0%	94.1% 5.9%
No Activity	**0** 0.0%	**0** 0.0%	**3** 3.8%	**20** 25.0%	87.0% 13.0%
	95.0% 5.0%	100% 0.0%	80.0% 20.0%	100% 0.0%	**93.8%** **6.3%**

Figure 4. 3: Confusion matrix demonstrating the final result with the best accuracy

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. The confusion matrix consists of rows and columns. The rows are the gesture classified by our system and the columns are the ground truth of a gesture [22]. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another) [23].

It is particularly important for our system to perform well on training data set of variable size. To test that, we fed our system with training set of size 1000 in total which consisted of 200 instances of each gesture. We found the accuracy to be 96% even when the training size was considerably reduced. Hence, these results show that our system is resilient and shows remarkable performance even when the data is scarce.

We have also used cross-validation to enhance the performance of our Deep Learning Algorithm and thus improve the accuracy. Cross-validation is a technique for evaluating Machine Learning models by training several machine learning models on subsets of the available input data and evaluating them on the complementary subset of the data. Cross-validation is used to detect problems like overfitting, i.e., failing to generalize a pattern [24]. The results clearly show that we get pretty high accuracy for identifying three main human gesture and also no gesture. Hence overall, the proposed system is highly accurate in recognizing different gestures.

5 Conclusion

The use of human gestures has become increasingly important in the fields of IoT for Smart Homes, caring services for seniors, security etc. The constructed model can be generalized and applied to different environments, making the system easily scalable. In this project we have shown that we can use Wi-Fi signals to detect human gestures like walking, standing, sitting, and no gesture (empty room) that can be used for developing a human monitoring system in Smart Homes. It has many advantages over device-based or camera-based approaches as it is cheaper and also does not require installing any additional devices. On the other hand, our system is device-free and low-cost that does not require any human participation. The semi-supervised based training approach detects human gestures at high accuracy that can be used over diverse fields. The results from our experiments suggest that this low-cost Wi-Fi-based system is highly reliable and scalable on monitoring of human gestures in indoor environments.

6 References

[1] Guo, Xiaonan, et al. “WiFi-Enabled Smart Human Dynamics Monitoring.” *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems - SenSys 17*, 2017, Press. https://doi.org/10.1145/3131672.3131692.

[2] A. Ng. [Online]. Available: https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf

[3] Autoencoders.[Online].Available:https://www.edureka.co/blog/autoencoders-tutorial/#applications

[4] Saandeep Depatla, Arjun Muralidharan, and Yasamin Mostofi. 2015. Occupancy estimation using only wifi power measurements. IEEE Journal on Selected Areas in Communications 33, 7 (2015), 1381–1393.

[5] Zhang, J. *Development and Implementation of Smart Home Monitoring System in Embedded Environment*. China.

[6] Softmax Regression.[Online].Available: https://docs.aws.amazon.com/machine-learning/latest/dg/cross-validation.html

[7] Martin Azizyan, IonutConstandache, and Romit Roy Choudhury.2009.Surround- Sense: mobile phone localization via ambience fingerprinting. In Proceedings of the ACM International Conference on Mobile Computing and Networking (ACM MOBICOM). 261–272.

[8] Saandeep Depatla, Arjun Muralidharan, and Yasamin Mostofi. 2015. Occupancy estimation using only wifi power measurements. IEEE Journal on Selected Areas in Communications 33, 7 (2015), 1381–1393.

[9] M Nakatsuka, H Iwatani, and J Katto. 2008. A study on passive crowd density estimation using wireless sensors. In Proceeding of the International Conference

[10] Siu-Yeung Cho, Tommy WS Chow, and Chi-Tat Leung. 1999. A neural-based crowd estimation by hybrid global learning algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 29, 4 (1999), 535–541.

[11] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. 2011. Tool release: gathering 802.11 n traces with channel state information. ACM SIGCOMM Computer Communication Review 41, 1 (2011), 53–53.

[12] Zhenjiang Li, Yaxiong Xie, Mo Li, and Kyle Jamieson. 2015. Recitation: Rehearsing wireless packet reception in software. In Proceedings of the ACM International Conference on Mobile Computing and Networking (ACM MOBICOM). 291–303.

[13] https://www.nia.nih.gov/health/publication/participating-activities-you enjoy. 2015. Participating in Activities You Enjoy. (2015).

[14] ChenshuWu,ZhengYang,ZimuZhou,KunQian,YunhaoLiu,andMingyanLiu. 2015. PhaseU: Real-time LOS identification with WiFi. In Proceeding of the IEEE Conference on Computer Communications (IEEE INFOCOM). 2038–2046.

[15] YanWang,JianLiu,YingyingChen,MarcoGruteser,JieYang,andHongboLiu. 2014. E-eyes: device-free location-oriented activity identification using fine- grained WiFi signatures. In Proceedings of the ACM International Conference on Mobile Computing and Networking (ACM MobiCom). 617–628.

[16] Chunmei Han, Kaishun Wu, Yuxi Wang, and Lionel M Ni. 2014. WiFall: Device-free fall detection by wireless networks. In Proceedings of the IEEE International Conference on Computer Communications (IEEE INFOCOM). 271–279.

[17] ("Train Stacked Autoencoders for Image Classification") Available: https://www.mathworks.com/help/deeplearning/examples/train-stacked-autoencoders-for-image-classification.html

[18] MATLAB App designer. [Online]. Available: https://www.mathworks.com/products/matlab/app-designer.html

[19] CSI Tools. [Online]. Available: https://dhalperi.github.io/linux-80211n-csitool/

[20] Machine Learning. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning

[21] Softmax Regression. [Online]. Available: http://ufldl.stanford.edu/wiki/index.php/Softmax_Regression

[22] Confusion Matrix. [Online]. Available: https://www.mathworks.com/help/stats/confusionmat.html

[23] Confusion Matrix. [Online]. Available: https://en.wikipedia.org/wiki/Confusion_matrix

[24] Cross-validation. [Online]. Available: https://docs.aws.amazon.com/machine-learning/latest/dg/cross-validation.html