

**Deep Learning-Based Stacking Neural Network and Generative Adversarial Networks for  
Human Activity Recognition Based on Ambient Sensors**

by

Qixuan Zhao

Submitted in partial fulfilment of the requirements  
for the degree of Master of Applied Science

at

Dalhousie University  
Halifax, Nova Scotia  
December 2020

© Copyright by Qixuan Zhao, 2020

## TABLE of Contents

<b>TABLE of Contents</b> .....	<b>ii</b>
<b>LIST OF TABLES</b> .....	<b>iv</b>
<b>LIST OF FIGURES</b> .....	<b>v</b>
<b>ABSTRACT</b> .....	<b>vii</b>
<b>LIST OF ABBREVIATIONS USED</b> .....	<b>viii</b>
<b>ACKNOWLEDGEMENTS</b> .....	<b>ix</b>
<b>Chapter 1: Introduction</b> .....	<b>1</b>
<b>1.1 Research Objective</b> .....	<b>3</b>
<b>1.2 Thesis Structure</b> .....	<b>4</b>
<b>Chapter 2: Literature Review of HAR</b> .....	<b>5</b>
<b>2.1 Types of HAR</b> .....	<b>6</b>
2.1.1 Camera-based HAR.....	6
2.1.2 Sensor-based HAR.....	7
<b>2.2 Deep Learning for HAR</b> .....	<b>9</b>
2.2.1 Fully Connected Neural Networks.....	12
2.2.2 Convolutional Neural Network.....	13
2.2.3 Long Short-Term Memory (LSTM).....	15
<b>2.3. Stacking</b> .....	<b>18</b>
<b>2.4 GANs for HAR</b> .....	<b>20</b>
<b>Chapter 3: Deep Learning-Based Stacking method for HAR Based on Ambient Sensors</b> .....	<b>23</b>
<b>3.1 CASAS Dataset for HAR Using Ambient Sensors</b> .....	<b>24</b>
<b>3.2 Generation of Base Models</b> .....	<b>26</b>
3.2.1 Architectures of Base Models.....	26
3.2.2 Training Data of Base Models.....	30
3.2.3 Sliding Window Size of Base Models.....	30
<b>3.3 Meta Model Based on LSTM</b> .....	<b>31</b>
<b>3.4 Pre-test</b> .....	<b>33</b>
3.4.1 Impact of Diversity of Architectures.....	33
3.4.2 Impact of Diversity of Training Data.....	35
3.4.3 Impact of Diversity of Sliding Window Size.....	36
3.4.4 Pre-test Conclusion.....	38
<b>3.5 Proposed Deep Learning-based Stacking Method for HAR Based on Ambient Sensors</b> .....	<b>39</b>
<b>3.6 Ambient Sensor Generative Adversarial Network</b> .....	<b>40</b>
<b>Chapter 4: Experiments and results</b> .....	<b>46</b>

<b>4.1 Deep Learning-Based Stacking Method for HAR Using Ambient Sensors.....</b>	<b>46</b>
<b>4.2 Ambient Sensor Generative Adversarial Network .....</b>	<b>50</b>
<b><i>Chapter 5: Conclusions and Future research .....</i></b>	<b>53</b>
<b><i>BIBLIOGRAPHY.....</i></b>	<b>56</b>

## LIST OF TABLES

Table 1 Mathematical equations and graphs of frequently used activation functions .....	11
Table 2 Statistical information about datasets used in the research.....	24
Table 3 Example of the raw data-based input.....	25
Table 4 The input of the meta model .....	31
Table 5 Wilcoxon signed-ranks test for BM10.....	49
Table 6 Wilcoxon signed-ranks test results regarding proposed stacking method.....	49
Table 7 Improvement (percentage points) made by ASGAN.....	50
Table 8 Wilcoxon signed-ranks test results of ASGAN .....	50

## LIST OF FIGURES

Figure 1 Mathematical model of an artificial neuron unit [18] .....	10
Figure 2 The structure of a typical fully connected neural network [18] .....	12
Figure 3 An example of a typical CNN .....	14
Figure 4 An example of how the convolution works in a CNN [34].....	14
Figure 5 An example of the mechanism of pooling [34] .....	15
Figure 6 An example of a LSTM network.....	15
Figure 7 A LSTM cell consists of three gates: forget gate, input gate, and output gate.....	16
Figure 8 Demonstration of the process of stacking [39].....	19
Figure 9 Generative adversarial networks [44].....	21
Figure 10 Sample data from Aruba dataset.....	25
Figure 11 The architecture of base model CNN-1 with four convolution and two fully-connected layers (FC). .....	27
Figure 12 The architecture of base model CNN-2 with eight convolution and two fully-connected layers.....	27
Figure 13 The architecture of base model CNN-3 with four convolution and four fully-connected layers.....	28
Figure 14 The architecture of base model LSTM-1 with one LSTM layer and two fully-connected layer .....	29
Figure 15 The architecture of base model LSTM-2 with two LSTM layer and two fully-connected layer .....	29
Figure 16 The architecture of base model LSTM-3 with one LSTM layer and two fully-connected layer .....	30
Figure 17 The architecture of the meta model .....	33
Figure 18 Box plot comparison of the accuracy of Model Benchmark, Model 1, and Model 2 ..	34
Figure 19 The result of Nemenyi test for group of Model Benchmark, Model 1, and Model 2...	35
Figure 20 Box plot comparison of the accuracy of Model Benchmark, Model 3, and Model 4 ..	36
Figure 21 The result of Nemenyi test for group of Model Benchmark, Model 3, and Model 4...	36
Figure 22 Box plot comparison of the accuracy of Model Benchmark, Model 5, and Model 6 ..	37
Figure 23 The result of Nemenyi test for group of Model Benchmark, Model 5, and Model 6...	38
Figure 24 Proposed ASGAN’s architectures. This model consists of two components: generator G and discriminator D.....	41
Figure 25 Generator of proposed ASGAN .....	42
Figure 26 Discriminator of proposed ASGAN .....	43

Figure 27 Results of experiments on three stacking model compared with baseline ..... 47

Figure 28 Comparison of accuracy between three pure data driven models and three mixed data models whose training data are enlarged by ASGAN ..... 51

## ABSTRACT

Smart home for healthcare services has acquired more attention since the increasing development of the Internet of Things and the population ageing over the world. Human activity recognition (HAR) is one of the concerns of the smart home. Ambient sensors based HAR is one promising direction. This research proposes a deep learning-based stacking method for HAR using ambient sensors. We first generate base models of convolutional neural networks (CNNs) and long short-term memory (LSTM) with different architectures, training data samples, and sliding window sizes. These base models are further integrated by a LSTM model to make final predictions. Furthermore, we propose a generative adversarial network to generate synthetic data as supplementary training data to tackle the problem of insufficient data. These two methods are used together on six real-world datasets. Results show that our proposed methodology statistically outperforms other approaches in the literature.

## LIST OF ABBREVIATIONS USED

ASGAN	Ambient Sensor Generative Adversarial Network
CNN	Convolutional Neural Networks
DL	Deep Learning
FCNN	Fully Connected Neural network
GAN	Generative Adversarial Network
HAR	Human Activity Recognition
HMM	Hidden Markov Model
kNN	k Nearest Neighbour
LSTM	Long Short-Term Memory
SDNN	Stacked Deep Neural Network
SVM	Support Vector Machine



## ACKNOWLEDGEMENTS

First, I would like to thank my supervisor, Dr. Alireza Ghasemi, for his kind help during my two years master study. His continued advice helped me learn the essential qualities and abilities to be a qualified MASc student. Working with him also motivates me to keep improving myself by learning from both the past failure and peer students. Without him, I would never be able to get this achievement. It is my luck to have Dr. Ghasemi as my supervisor.

Hanna Lo is appreciated as well. She helped me a lot by giving me suggestions on how to organize my thesis and polish the language.

I would also like to thank my family, my parents, my sister, and my aunt Xia, for their unconditional support and understanding that gave me the opportunity and courage to pursue further study.

## Chapter 1: Introduction

Population ageing, which is now widespread globally, is a shift in the distribution of a country's population towards older ages. Canada is no exception to this phenomenon. Canada Statistics projected that the proportion of seniors (those aged 65 and older) is about to reach at a minimum of 22.7% by 2033 in Canada. In addition, according to the Canadian Community Health Survey, the portion of the elderly with at least one chronic body condition has exceeded 50% and will continue to increase. The trend towards population ageing and the increasingly high proportion of seniors with chronic body conditions have both contributed to the rise in the demand for home healthcare. In 2017, the industry of home healthcare services in Canada was up 6.1% from the previous year. Among the various home health care services, smart home for elderly care is the fastest growing one, given it is expected to show an annual growth rate of 18.2% between 2020 and 2023. Smart Home for elderly care aims to provide an environment where the elderly can live independently and safely while relieving caregivers' stress through monitoring the elderly's daily activities and providing feedback to caregivers. A key aspect of smart home for elderly care is the feedback provided to the caregivers, which can be records of resident's daily activities events or alarms triggered by abnormal behaviours (e.g., coma, fall). One method for collecting this feedback is Human Activity Recognition (HAR). HAR uses various equipment to detect and recognize residents' activities, collect that information, and process and analyse them later. With HAR's help, the elderly and their caregivers can track the user's living habits to help improve quality of life and health conditions [1].

Based on the devices used to record residents' activities, there are two general HAR types: camera-based HAR and sensor-based HAR. The camera-based method uses cameras to collect vision-based data containing information about human activities. Although the camera-based

HAR benefits a lot from the fruitful information in the visual data, it still has drawbacks such as expensive computation cost, high dependence on the quality of the foreground segmentation, and the resident's concern for the privacy, which has made sensor-based HAR more popular [2]

Sensor-based HAR uses various sensors to collect data generated by residents' activities.

Therefore, it does not require as much computation resources nor invades privacy as much as the camera-based method. Sensor-based HAR can also be split into two categories: wearable sensors or ambient sensors. Wearable sensors are placed on the resident's body and collect postural and locomotive data of the user. On the other hand, ambient sensors are placed in the living environment and record the resident's daily activity information such as location and appliance being used. Ambient sensors typically include motion sensors, switch sensors, and pressure sensors. Although these categories are both feasible, users have shown resistance to wearing sensors on their bodies continually [3]. Consequently, HAR based on ambient sensors become increasingly promising [4].

The core of HAR is to build a model for recognizing residents' activities with recognition accuracy. Various techniques have been applied to this problem of HAR, including, but not limited to, hidden Markov model (HMM), support vector machine (SVM), k nearest neighbour (kNN), and deep learning (DL). A review of the current literature on these techniques will be discussed in Chapter 2. Among these techniques, deep learning is becoming increasingly popular because of its ability to simultaneously learn features extraction and classification [5]. Also, stacking is a technique of interest frequently used to further improve classification performance, given its ability to improve the generalization capabilities of models by combining multiple classifiers. The details of the stacking will be further explained later.

On the other hand, most HAR research merely focuses on improving models' recognition accuracy and ignoring the data itself [6]. As a matter of fact, HAR's performance highly relies on both the quantity and the quality of the collected data. However, it is getting increasingly difficult to obtain enough data for HAR due to two facts [7]. The first is that collecting HAR related data in a smart home is time-consuming because it takes months to collect enough data to make HAR models learn the user's living habits efficiently. The second is that it is expensive to label activities on the data collected from sensors. Hence, the lack of sufficient data caused by high costs on time and money becomes another challenge that HAR based on ambient sensors needs to tackle.

### 1.1 Research Objective

The main objective of this research is to propose an improved deep learning-based stacking method for HAR based on ambient sensors. This includes proposing an approach of generation and an algorithm of integration. To achieve this objective, three phases of study will be carried out. The first is to explore feasible factors whose diversities can have positive impacts on the performance of stacking model. The next phase is to construct a deep learning-based model for integration. The third phase is to propose a stacking based algorithm amid HAR using ambient sensors.

The second objective of this research is to solve the problem of insufficient data. A model based on generative adversarial networks (GANs), whose detail will be included in Chapter 2 and Chapter 4, is introduced. This model is called ASGAN (ambient sensor GAN). We will use ASGAN to generate synthetic data and further mix the synthetic data with real data collected by ambient sensors to train our proposed HAR model. To the author's best knowledge, this is first work on implementing GAN in the area of HAR based on ambient sensors.

## 1.2 Thesis Structure

The rest of this thesis is organized as follows:

- Chapter 2 provides a current review on literature related to the HAR problem, including different types of HAR, its application, and various classification methods having been applied. Besides, literature reviews on the application of both stacking and GAN are also included.
- Chapter 3 proposed a deep learning-based stacking method for HAR based on ambient sensors.
- Chapter 4 proposed a GAN-based model, called ASGAN, aiming to generate synthetic ambient sensors data to tackle the lack of sufficient data and further improve the performance of HAR.
- Chapter 5 presents the experiment and the corresponding experimental evaluation and result based on six ambient sensors based datasets for HAR.
- Chapter 6 includes the conclusion and discussion of this research, as well as some suggestions for future work.

## Chapter 2: Literature Review of HAR

Human activity recognition (HAR) aims to determine residents' activities inside smart homes with the help of various devices. Such devices include cameras, wearable sensors, and ambient sensors. Based on the devices used to record residents' activities, there are two general HAR types: camera-based HAR and sensor-based HAR. The difference between these two types of HAR is in the data type used, which leads to different pre-processing feature extraction methods. The rest of this Chapter will bring more detail about these two different types of HAR.

No matter what devices are used and regardless of which HAR type, HAR typically involves three steps [8]: pre-processing, feature extraction, and classification.

1. Pre-processing: collect raw data from devices and organize them into a certain form for further utilization.
2. Feature extraction: extract features containing useful information from raw data to construct the input.
3. Classification: use a trained model to determine the given activity based on data derived from devices.

In pre-processing, raw data is first collected from the devices. Then the data is organized into a certain form or processed by certain algorithms for further utilization. For example, firstly, videos are broken down into frames of pictures. Then human are further extracted from the background. What occurs during feature extraction influences the HAR's performance greatly as it involves extracting useful features from the raw data, filtering ineffective information, and reducing the computation cost [9].

## 2.1 Types of HAR

### 2.1.1 Camera-based HAR

The camera-based method uses cameras to collect vision-based data containing information about human activities. These data are usually videos that record residents' daily lives.

Before HAR methods can analyze the human activities in these videos, the videos will need to be broken down into frames, and foreground segmentation or background subtraction will need to be conducted [10-15]. In other words, the resident and the background must be separated. This step can also reduce the computation cost required by further analysis. [12] used the mixture-of-Gaussian model to classify each pixel based on whether the Gaussian distribution representing it most efficiently is considered part of the picture's background. However, this method does not work well when the background objects are clustering and do not stay wholly static. To tackle this problem, [13] proposed the codebook background subtraction algorithm, which has an advantage in dealing with a moving and clustering background. Instead of using traditional RGB cameras, [14] and [15] both used depth-sensing cameras. By doing so, they can directly track the user's joints without separating humans and backgrounds and conduct HAR more efficiently. However, only tracking the user's joints ignored information contained in the background objects. As a result, methods relying merely on tracking joints cannot handle human activities, which are the interaction between humans and backgrounds. On the other hand, [50] employed a hybrid model consisting of four CNNs. These CNNs were fed by different data: one CNN is based on RGB images, the other three CNNs are based on depth images of three different dimensions. This hybrid model utilizes information contained in both RGB images and depth images.

Although camera-based HAR benefits a lot from the visual data's fruitful information, it still has drawbacks such as expensive computation cost, a strict requirement for the environment, and the resident's concern for privacy [2]. The visual data itself cause an expensive computation cost. Like what was mentioned above, video streams typically will be first preprocessed. Then data after preprocessing will be processed to recognize activities by specific algorithms. Both preprocessing and recognition are required to be conducted in real-time, which will require much computation, especially given the large size of multiple image-base data nowadays. Camera-based HAR requires a strict smart home environment to make sure the normal function of recognition algorithms. For example, when occlusions, where a room contains furniture or objects placed between the person and the camera, happened, algorithms may make inaccurate recognitions. Other factors, such as low light and changes in the layout of furniture, could also negatively influence the performance of camera-based HAR. Privacy concern is another problem. Given that the camera will collect visual data, including some sensitive information such as the user's appearance, users show vital concern for their privacy and the unpleasant feeling of "being watched" [2].

#### 2.1.2 Sensor-based HAR

Like what was mentioned above, camera-based HAR faces challenges including expensive computation cost, strict requirement for the environment, and the resident's concern for the privacy. However, sensor-based HAR does not require as much computation resources nor invades privacy as much as the camera-based method. As a result, sensor-based HAR is more popular in the HAR area for smart home residents [2].

Sensor-based HAR can also be split into two categories: wearable sensors or ambient sensors. Wearable sensors are placed on the resident's body and collect postural and locomotive data of the user. On the other hand, ambient sensors are placed in the living environment and record the



resident's daily activity information such as location and appliance being used. Ambient sensors typically include motion sensors, switch sensors, and pressure sensors. Of the two types of sensor-based HAR, ambient sensors are more popular because users have shown resistance to wearing sensors [4].

Given the drawbacks mentioned earlier of the camera-based method and wearable sensors based method as well as the advantages of ambient sensors based method, this project will henceforth focus on ambient sensor-based HAR.

Various conventional machine learning methods have been successfully used to analyze the activities recorded by ambient sensors. However, all of them faces two challenges: feature extraction and heterogeneity [16].

Firstly, the quality of feature extraction has great impacts on the performance of classifiers [5]. However, conventional machine learning methods require manual feature extraction and hence prior knowledge. Such prior knowledge helps to select the appropriate features. Let us consider HAR based on ambient sensors. First, sensor activation may reflect the activities that the resident is doing. If the sensor on the toilet is activated, then it is obvious that the resident is using the bathroom. Therefore, we may consider the activation status of sensors as a feature. The duration and the sequence of the sensors' activation may also help us to identify the activity. If the motion sensors in the kitchen and the restroom were activated successively but only lasting for a few seconds and then the motion sensor in the bedroom was activated, it may indicate that the resident was passing through the kitchen and restroom to go to the bedroom. Hence, the duration and the sequence of the sensors' activation may also be considered useful features. In the above examples, the activation status of sensors and the corresponding duration and the sequence are selected as features because we have prior knowledge that these features are useful. However,

given the complexity of the sensor network, some prior knowledge is hard to get. As a result, some features that are helpful to construct effective classifiers may be ignored. Also, conducting feature extraction manually is a poorly generalizable endeavour [5]. In other words, some of these hand-crafted features are not distinguishable enough to help build accurate classifiers [17]. Besides, some of those features extracted from raw data might be redundant and irrelevant, which is another challenge HAR has to face [9]. In the worst cases, such unhelpful features could even have negative impacts on the performance.

Secondly, users and sensors layouts are the two most important factors of HAR. Different users may have diverse activity styles. On the other hand, different sensors layouts may generate significantly different sensory data. As a result, different users and sensors layouts can lead to the heterogeneity of the sensory data for activity recognition, which may cause drops in the recognition accuracy when the classifiers are not design for specific users and sensors layouts. Given these two challenges of feature extraction and heterogeneity, deep learning becomes an increasingly popular in recent years because it doesn't need manual feature extraction and is flexible. Because of deep learning's layer-by-layer structures, it can not only simultaneously learn data features and classifiers but also be flexibly unified to deal with sensory data generated by different users and sensors layouts [5,16], which will be further explained in the next section. Given deep learning's potential to deal with challenges of feature extraction and heterogeneity, it is worthy of paying more attention to deep learning. The following section will provide a brief explanation of deep learning.

## 2.2 Deep Learning for HAR

Deep learning is a general term for Artificial Neural Network (ANN) algorithms consisting of multiple hidden layers. Such layers are made up by multiple artificial neuron cells.

For an artificial neuron cell, its mathematical model can be demonstrated as Figure 1. In Figure 1, signals sent by other artificial neuron cells are  $x_i$ . After being received, signals can be represented as the product of  $x_i$  and  $w_i$ . In the cell body, all received signals are add as  $\sum_i w_i x_i + b$ , where  $b$  is the bias of the current cell, and are further processed by an activation function  $f$ .

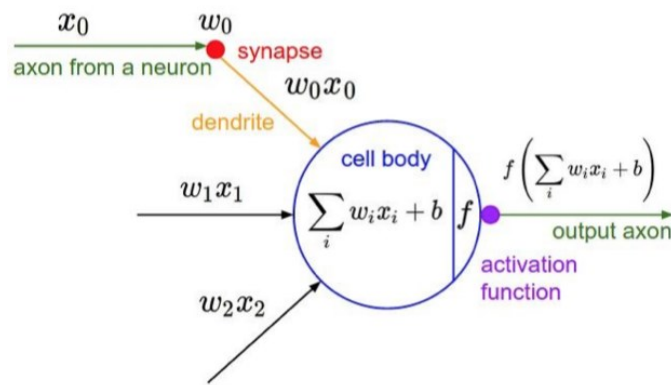
$$y = f(\sum_i w_i x_i + b) \tag{1}$$


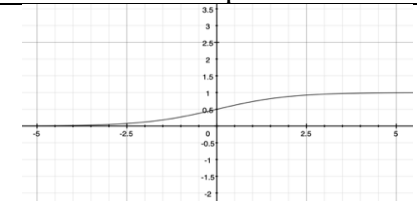
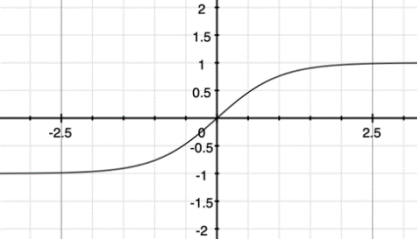
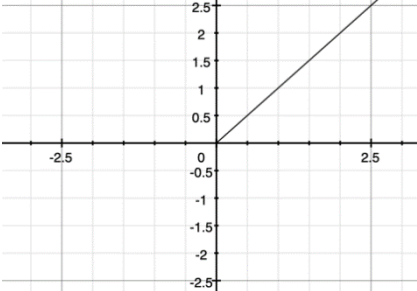

Figure 1 Mathematical model of an artificial neuron unit [18]

This whole process can be represented as Equation 1, where  $y$  is the signal outputted by the artificial neuron cell. Table 1 shows four different activation functions: sigmoid, tanh, relu, and softmax.

For sigmoid activation, shown in Table 1, the value of output takes the value between 0 and 1. This range could be interpreted as the activation rate of the neuron, where 0 represents a neuron that is not activated at all, and 1 represents a fully activated neuron. Output of tanh takes the value between -1 and 1. Its output can also be considered as the activation rate of the neurons, where negative values represent inhibited neurons, and positive values represents activated neurons. Sigmoid and tanh are frequently used when the output is required to have both upper and lower limits like images. For instance, every pixel in an image can be expressed by a value between 0 and 1, or -1 and 1. On the other hand, unlike sigmoid and tanh, relu activation does

not have upper limit. When the inputted value is equal or less than 0, it won't be activated. Relu is frequently in layers before the last output layer when there are no certain requirements for these layers' outputs, given relu has the advantage that its gradient won't vanish during training [19]. The last activation function in Table 1 is softmax. The output of softmax activation is a vector of k values that are between 0 and 1. Also, such k values sum up to 1. Therefore, the output of softmax can be considered as a probability distribution with k classes. Softmax is often used as the output layers in classification problems. More detail of them will be explained in Chapter 3.

Table 1 Mathematical equations and graphs of frequently used activation functions

Name	Mathematical Equation	Graph
sigmoid	$f(z) = \frac{1}{1 + e^{-z}}$	
tanh	$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	
relu	$f(z) = \max(0, z)$	
softmax	$f(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}}$	

Deep learning has various architectures including fully connected neural networks (FCNN), convolutional neural networks (CNNs), and long short-term memory (LSTM). The following section will bring more detail about these three different architectures.

### 2.2.1 Fully Connected Neural Networks

Shown as Figure 2, a FCNN consists of layers where all neurons in a certain hidden layer are fully connected to the previous layer and the next layer. In a such network, hidden layers learn how to extract features and classify. FCNN has been applied to various areas such as agriculture, medicine, and finance [20-27]. In agriculture, [20] used a FCNN for fruit classification where texture and shape features of each fruit image are taken as the input. [21] utilized FCNNs to predict crop yield by taking as input parameters like environment temperature, rainfall, humidity etc. FCNN has also been applied to the medical area. [22-24] utilized FCNN to analyze patient’s symptom data to make diagnosis. In finance, [25,26] utilized the stock price with the help of FCNNs. Also, [27] constructed a FCNN to predict future failure of finance company according to the company’s financial report.

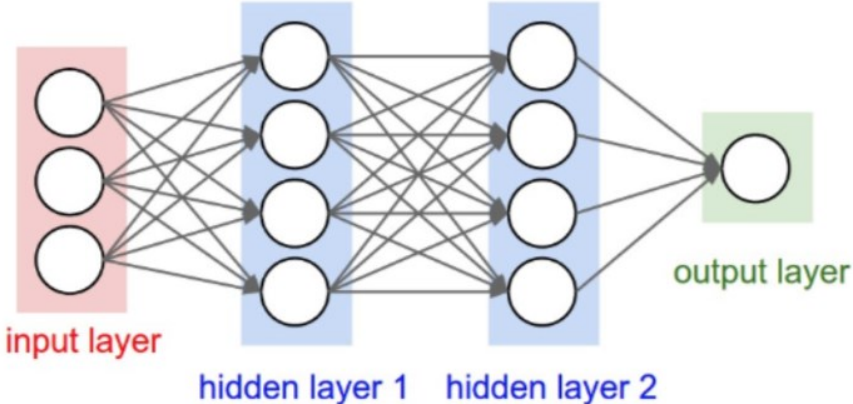


Figure 2 The structure of a typical fully connected neural network [18]

### 2.2.2 Convolutional Neural Network

Convolutional Neural Network (CNN), inspired by the architecture of the visual cortex, is a model resembling the connectivity pattern of neurons in the animal's visual cortex. Figure 3 provides an example of a typical CNN, where *conv* refers to convolution, *pool* refers to pooling, *FC* refers to fully connected. Unlike FCNN, where every neuron is fully connected to all the neurons from its previous layer and its next layer, each neuron in the convolution layer is only connected to neurons of a specific area from the previous layer. In other words, each neuron in the convolution layer is the convolution output of the previous layer's neurons in a particular area. Therefore, CNN gains the ability to capture spatial and temporal information by conducting convolutions in every input area. Take the CNN in Figure 3 as an example, this CNN model has convolution layers and pooling layers. The convolution layers learn data representation while the pooling layers, following these convolution layers, play the role of down sampling. After learning feature extraction, fully-connected layers placed after the convolution and pooling layers fulfill the job of classification. As a result of the convolution layers and fully-connected layers, the CNN gains the ability to simultaneously learn feature extraction and classifiers. To have a better understanding of how convolution works in CNNs, consider Figure 3. The black blocks in Figure 3 represent convolution kernels which move through the entire grey maps to extract features. Figure 4 provides a concrete worked out example of this convolution process. The area in the khaki color outlines the current position of the kernel as it slides through the input doing element-wise multiplication and addition. The element-wise multiplication and addition is represented by Equation 2, where  $y$  refers to the result of the convolution, and  $x$  is the elements of the input where the kernel is currently positioned, and  $w$  is the weight of the kernel. Using the equation, the result of convolution in the current khaki area, in Figure 4, is 429 ( $18*1+54*0+51*1+55*121*1+75*0+35*1*24*0+204*1$ ).

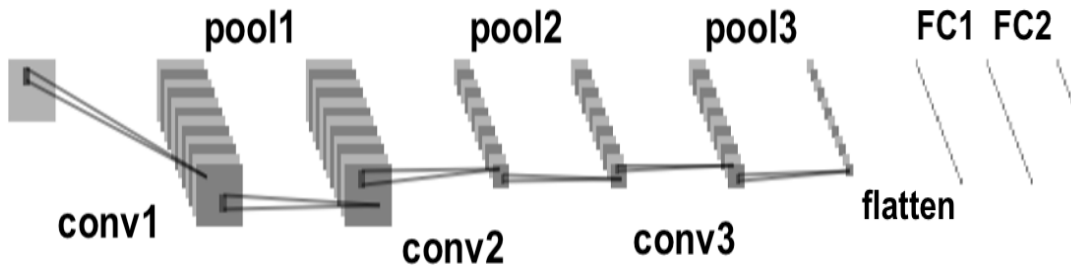


Figure 3 An example of a typical CNN

$$y = \sum_i \sum_j w_{ij} x_{ij} \quad (2)$$

Like convolution, pooling is also conducted by a filter. Shown in Figure 5, there are two commonly used types of pooling: max pooling and average pooling: max pooling returns the maximum value covered by the pooling filter, and average pooling returns the average value. There is no guarantee that which one of these types of pooling work generally better than the other. But they all aim to reduce the size of the features extracted by the previous convolution layer. After the convolution layers and pooling layers learn the features, the extracted features are flattened into a vector from the matrix of a higher dimension, shown in Figure 3. The extracted features are flattened so they can be processed by the fully connected layer to finish the classification job.

INPUT					
18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

WEIGHT		
1	0	1
0	1	0
1	0	1

429
-----

Figure 4 An example of how the convolution works in a CNN [35]

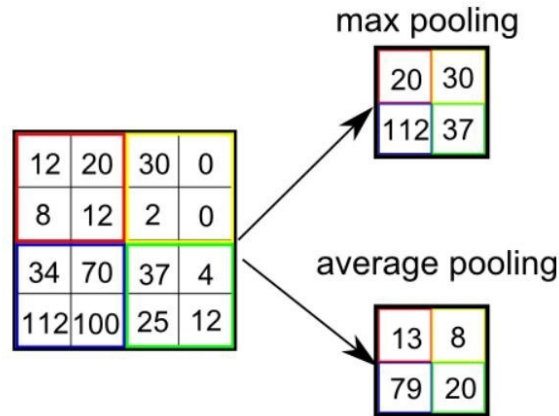


Figure 5 An example of the mechanism of pooling [35]

### 2.2.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is another variant of deep learning. Shown in Figure 6, a typical LSTM features multiple LSTM layers. Such LSTM layers process current moment's input  $x_t$  with considering not only  $x_t$  itself but also the last moment's output  $h_{t-1}$ . A LSTM layer is made up by LSTM cells whose architectures are shown in Figure 7. The LSTM cell has three special gates: forget gate, input gate, and output gate. The information flow inside LSTM cell first go through forget gate, then input gate, and finally output gate.

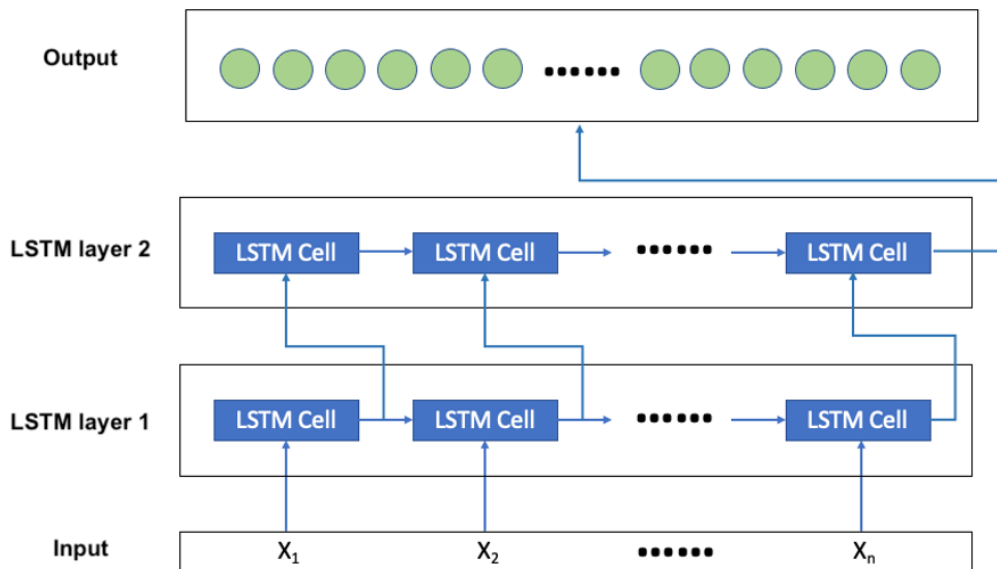


Figure 6 An example of a LSTM network



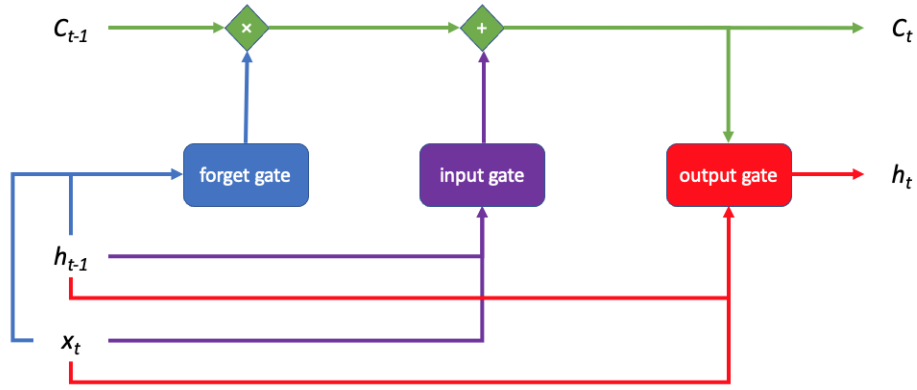


Figure 7 A LSTM cell consists of three gates: forget gate, input gate, and output gate.

Inside a LSTM cell, shown in Figure 7, the first step is to decide what information contained in last moment's cell state  $C_{t-1}$  should be forgotten. The Cell state  $C_{t-1}$  here is the information stored in the LSTM cell at the last moment  $t-1$ . This decision of which parts of information the cell should forget is made by forget gate. The forget gate uses output of last moment  $h_{t-1}$  and current input  $x_t$  to output a forget rate  $f_t$  between 0 and 1, where 0 represents that the information contained in  $C_{t-1}$  is completely forgotten and 1 represents that the information contained should be fully accepted. The mathematical model of forget gate is represented by Equation 3, where  $W_f$  and  $b_f$  are the weight and bias of forget gate. The values of  $W_f$  and  $b_f$  are gained during training. Also,  $\sigma$  here is the sigmoid activation shown in Table 1. Therefore, the information reserved by the forget gate is  $f_t * C_{t-1}$ .

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3)$$

After forget gate is input gate, which outputs the current cell state  $C_t$ . Equation 4 is the mathematical model of input gate, where  $W_i$  and  $b_i$  are the weight and bias of input gate,  $W_C$  and  $b_C$  are the weight and bias of the LSTM cell,  $\tanh$  is the tanh activation function shown in Table 1. In input gate,  $\sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$  is the information  $i_t$  gained from the output of last step  $h_{t-1}$  and current input  $x_t$ . And  $\tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$  is the input rate between -1 and 1,

where -1 represents that  $i_t$  should be completely eliminated from the information reserved by the forget gate,  $f_t * C_{t-1}$ , while 1 represents the opposite situation.

$$C_t = f_t * C_{t-1} + \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) * \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (4)$$

Next, the updated current cell state  $C_t$ , gained by input gate, will be stored in the cell for further use at the next moment. Also,  $C_t$  will be passed to output gate to generate current moment's output  $h_t$ . Output gate takes  $h_{t-1}, x_t$ , and  $C_t$  as inputs to generate  $h_t$  by using Equation 5, where  $W_o$  and  $b_o$  are the weight and bias of output gate.  $\sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$  is the output rate that decides what parts of current cell state  $C_t$  we are going to output. Also, it should be noticed that cell state  $C_t$  in output gate is regulated to values between -1 and 1 by using tanh activation. The motivation for using tanh here is that tanh activation's gradient computation is less expensive than other activation functions.

$$h_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) * \tanh(C_t) \quad (5)$$

In summary, a LSTM cell uses forget gate to control what parts of last moment's cell state  $C_{t-1}$  can be reserved. Next, input gate is utilized to output current cell state  $C_t$  by adding information from last moment's output  $h_{t-1}$  and current input  $x_t$  to the reserved parts of  $C_{t-1}$ . Finally, output gate generates current output  $h_t$  by taking  $h_{t-1}, x_t$ , and  $C_t$  as inputs. It can be seen that, besides  $x_t$ , the cell state  $C_{t-1}$  and  $h_{t-1}$  also run through the entire LSTM cell. In other words, LSTM cells generate the output by considering the last moment's information all the time. LSTM can always memorize what it has learned from previous moments and then pass such memories to the next moment. This is the reason why LSTM is good at time series data.

Applications of LSTM are varied, including but not limited to application in robotics, speech recognition, and traffic forecasting. [28] used human-generated examples to train a robotic surgical manipulator on how to do knot winding trajectories with the help of LSTM. [29] applied

LSTM to speech recognition. In their work, they compared the bidirectional LSTM, where information could pass through in two directions (from  $X_1$  to  $X_t$ , and from  $X_t$  to  $X_1$ ), and unidirectional LSTM, where information can only be processed in one direction from  $X_1$  to  $X_t$ , and determined that the bidirectional LSTM had a slight advantage. Motivated by the availability of ambulant traffic data and the rapidly increasing computation power in recent years, [30] proposed an LSTM model for short-term traffic forecasting. Besides the above applications, there are still many fields that have applied LSTM such as handwriting recognition [31], business process management [32], prediction in clinical events [33], and airport passenger management [34].

### 2.3. Stacking

Stacking is a technique used to improve the classification's performance by combining multiple classifiers. Shown in Figure 8, stacking first generates and trains multiple base models. Then a new training data set is made from these base model predictions and used to train a meta-model to make final predictions. The process of generating the base models is called generation and the process of training the meta model to use the base models' outputs to make predictions is called integration. Generation and integration are two key considerations of the model stacking technique.

1. Generation: generate multiple classifiers, or base models, and make predictions.
2. Integration: make the final prediction based on outputs generated by base models

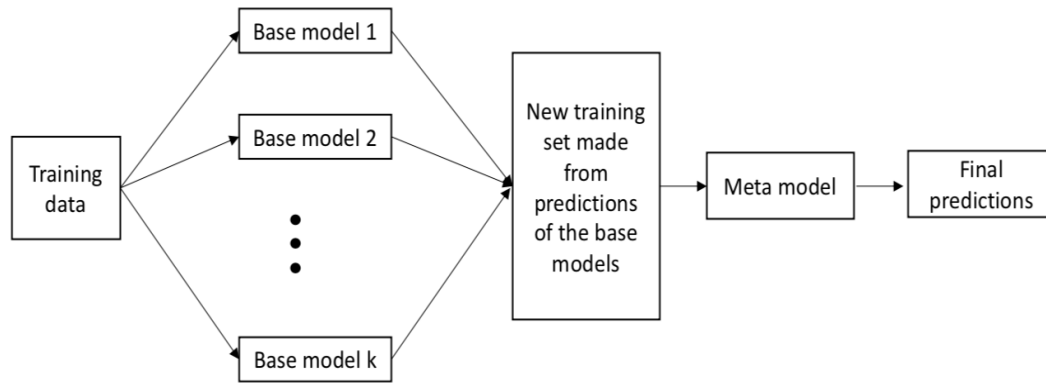


Figure 8 Demonstration of the process of stacking [40]

The goal of generation is to guarantee the diversity of base models. Without such diversity, the model stacking technique is not capable of improving the performance because the same data are always processed the same way [35]. Researchers have tested several methods for guaranteeing diversity, including diversifying training datasets, feature extraction methods, model architecture, and classification algorithms. [36] guaranteed base model diversity by training multiple base models on different training datasets compiled from data collected using three different wearable sensors placed on the chest, wrist, and ankle. [37] guaranteed diversity in the base models by conducting different feature extraction methods to extract different features for training the base models. [38] combined the above approaches. They first constructed multiple datasets by combining data collected by different sensors. Then different features are extracted from above data. Such features were further used to feed the base models. In addition to focusing on the diversity of training dataset or feature extraction methods, another direction for diversifying base models is building base models with different architectures. [39] varied the architecture by building three CNNs with different kernel sizes in both the convolution layers and pooling layers. Instead of generating base models with only one algorithm, [16] diversified the architecture by applying multiple classification algorithms.

For integration, the goal is to utilize the outputs from the base models efficiently. There are two general methods for integration. The first is the most commonly used voting method where the predictions for each class by base models are summed, and the class with the majority vote will be the final prediction. The second is to build another classifier that calculates a final prediction using the outputs of the base models as input.

Stacking has gained more popularity in recent years as a method aimed at improving the performance [4]. Considering model stacking's strong ability of enhancing model's performance and deep learning's great power of learning data representation, [4, 39] have even combined deep learning and model stacking to solve wearable sensor based HAR problems. Both researches show that the combined model outperforms previous state-of-the-art results. However, to the knowledge of the author, there has been no research completed that combines deep learning and model stacking for HAR in a smart home environment with ambient sensors.

#### 2.4 GANs for HAR

For HAR, collecting trainable data is a difficult task. For one thing, there should be volunteers all the time in order to monitor residents' activities and then label every sensor events. For another thing, residents are required to be cooperative during the stage of collecting data. This would bring inconvenience to residents since their daily life is needed to be monitored during a given period. As a result, collecting trainable data has become one of the challenges that ambient sensor-based HAR needs to tackle. As a matter of fact, this problem is not limited to ambient sensor based HAR. Wearable sensor based HAR also faces such a challenge. More recently, with the development of generative adversarial networks (GANs), researchers begin to use GANs to tackle the insufficient data problem.

As shown in Figure 9, a GAN consists of two networks: generator  $G$ , which generating synthetic data, and discriminator  $D$ , which distinguishes between real data and synthetic data. During the

training process, the generator's goal is to obtain a distribution that is close enough to the distribution of real data by improving the quality of the synthetic data. On the other hand, the discriminator continues to enhance its ability to distinguish between real data and synthetic data.

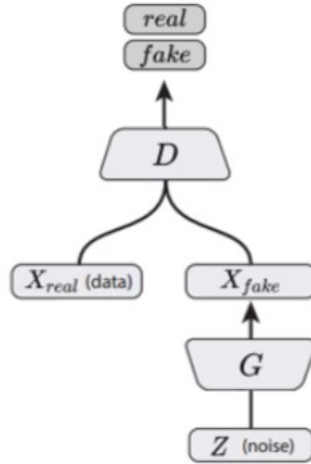


Figure 9 Generative adversarial networks [45]

The objective of GAN is to improve the generator's ability of generating synthetic data and enhance the discriminator's ability of distinguishing synthetic data simultaneously. Assume that  $x \sim p_{data}$  is a set of samples from the set of real data, then  $E_{x \sim p_{data}}[\log D(x)]$  is the logarithmic probability of real data  $x$  being recognized as real by discriminator. Hence, the goal of the discriminator is to maximize this logarithmic probability. On the other hand, assume  $z \sim p_z$  is samples of noise, then  $E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$  is the logarithmic probability that synthetic data, generated from noise  $z$  by generator  $G$ , being recognized as fake by discriminator  $D$ . Therefore, the goal of the generator is to minimize this probability. The objective of GAN can be expressed as Equation 6. It can be seen this objective requires GAN to simultaneously optimize both generator and discriminator by minimizing and maximizing the two types probability, which are mentioned above, respectively.

$$\min_G \max_D E_{z \sim p_z(z)} [\log(1 - D(G(z)))] + E_{x \sim p_{data}} [\log D(x)] \quad (6)$$

[6] is the first work to apply GANs to wearable sensor based HAR. They built different GAN models for every single activity. They used such GAN models to generate synthetic data and mix them with real data to train HAR models. Although this method increased the recognition accuracy, it requires adjusting the GAN model's architecture for every activity class. [6] built a GAN model that can handle all activity classes without further adjusting the architecture. By having such uniformed architecture, this GAN model can work more efficiently by saving time from adjusting architectures. Therefore, so far there are two ideas for applying GAN: build multiple GANs for different activities [6] or built one mere GAN [7]. Inspired by these two ideas, [18] proposed a hybrid method. They designed two GAN models: LSTM-based and CNN-based. Then activities were split into two types: relatively long-term static activities (like sit and still) and faster-changing activities (like run). By comparing the performance of LSTM-based GAN and CNN-based GAN on these two types of activities, [18] concluded that LSTM-based GAN performs better on relatively long-term static activities, and CNN-based GAN was more suitable to generate synthetic data of other activities.

To the author's best knowledge, there is currently no research on applying GANs to ambient sensor-based HAR. Given the insufficient data problem's negative influences on classifiers' performance and the success of applying GANs to oversample wearable sensor based HAR, it is worthy of working on GANs for ambient sensor based HAR.

## Chapter 3: Deep Learning-Based Stacking method for HAR Based on Ambient Sensors

For HAR using ambient sensors, one of the most important tasks is how to achieve high prediction accuracy [7]. Stacking is one of the technique most frequently used. It aims at improving machine learning models by combining multiple models. It involves two steps: generation and integration. For generation, multiple base models are first trained. Then their predictions are used to construct a new training set for the meta model. For integration, a meta model is trained by the new training data to learn how to avoid mistakes made by base models and hence make final predictions. There are two important considerations in these two steps: the diversity of base models and the algorithm of the meta model. Firstly, without diversity, base models will tend to make same mistakes in the way predicting certain data. As a result, the meta model cannot recognize this problem and learn how to improve the prediction by using base models that are better suited for that certain data. Also, it is important to note that base models must achieve relatively acceptable accuracy [40]. In other words, base models should maintain diversity and guarantee accuracy at the same time. Secondly, the algorithm of the meta model decides how predictions of base models are processed to make the final decision. Such algorithm should be able to learn how to make predictions using its base models' predictions.

This chapter will propose a method for ambient sensors based HAR combining stacking and deep learning. There are two motivations behind this. The first is the accuracy requirement for base models in the first consideration mentioned above. As a technique having strong ability of modelling complex, non-linear relationships, which is valuable for application in the HAR domain [41], deep learning is suitable to be chosen as the base models that can guarantee the required relatively acceptable prediction accuracy. Secondly, deep learning has strong abilities of



learning feature extractions and classification simultaneously. This makes deep learning also suitable to be meta model, given the core of integration is to use meta model to learn experiences from mistakes made by base models and then make final predictions.

### 3.1 CASAS Dataset for HAR Using Ambient Sensors

Before jumping into the proposed method, the detail of CASAS dataset we used in this research will be first provided to help have a better understanding of the data for HAR using ambient sensors. The CASAS project [42] treats the smart home as intelligent agents that can learn and use knowledge to detect residents’ activities. In such environments, the status of the residents and their physical surroundings are perceived using ambient sensors. The CASAS project datasets consist of data collected by ambient sensors in different smart homes. We used six of the datasets in the CASAS project for testing the proposed model. The statistical information, which includes the number of sensor events, sensors, and activities, for each of these datasets we used in the research can be found in Table 2. Figure 10 shows a sample of the Aruba dataset, an example of one of the CASAS project datasets. The dataset consists of data for every sensor event’s information, including the timestamp, sensor ID, and the activity label. For instance, the first line, or the first sensor event, in Figure 11 provides such information: On the day of 2010-11-04, sensor M004 was “ON” at the moment of 5:40:51.303739, and the activity of “Bed\_to\_Toliet” began.

Table 2 Statistical information about datasets used in the research

Dataset	Number of Sensor Events	Number of Sensor	Number of Activities
Aruba	1719558	39	11
Twor	137789	71	13
Cario	726534	32	13
Tulum	486912	20	9
Milan	433665	33	15
hh101	326066	76	31

```

2010-11-04 05:40:51.303739 M004 ON Bed_to_Toilet begin
2010-11-04 05:40:52.342105 M005 OFF
2010-11-04 05:40:57.176409 M007 OFF
2010-11-04 05:40:57.941486 M004 OFF
2010-11-04 05:43:24.021475 M004 ON
2010-11-04 05:43:26.273181 M004 OFF
2010-11-04 05:43:26.345503 M007 ON
2010-11-04 05:43:26.793102 M004 ON
2010-11-04 05:43:27.195347 M007 OFF
2010-11-04 05:43:27.787437 M007 ON
2010-11-04 05:43:29.711796 M005 ON
2010-11-04 05:43:30.279021 M004 OFF Bed_to_Toilet end

```

Figure 10 Sample data from Aruba dataset

To make this information usable, the data in the datasets is transformed and recorded in the form of a matrix. Assuming that we will trace back  $M$  sensor events each time when we try to recognize current sensor's activity and there are  $N$  sensors in this smart home, then the matrix is represented in Table 3, where  $T_M$  is the  $M_{th}$  sensor event, the first  $N$  columns records the status of  $N$  sensors at any given sensor event. And the number of sensor events being traced back is defined as the sliding window size. In Table 3, the sliding window size is  $M$ . Also,  $x_{M,N}$  takes values of -1 or 1. When sensor  $N$  is activated at sensor event  $M$ ,  $x_{M,N}$  takes the value of 1. When sensor  $N$  is turn off, then  $x_{M,N}$  takes the value of -1. Otherwise,  $x_{M,N}$  will be 0.

Table 3 Example of the raw data-based input

	Sensor 1	.....	Sensor N
$T_1$	$X_{1,1}$	.....	$X_{1,N}$
.....	.....	.....	.....
$T_M$	$X_{M,1}$	.....	$X_{M,N}$

By transforming the records of sensor events, shown as Figure 10, into the format demonstrated as Table 3, the input can not only contain the spatial information, the status of the sensor system, but also include the temporal information, the timestamp and the sequence of sensor events.

Considering the spatial and temporal information included in the dataset and CNNs strong ability

to deal with spatial information and LSTMs ability to handle sequence processing tasks, we use these models as our base models.

### 3.2 Generation of Base Models

Like what mentioned before, the generation of base models is one of the vital parts of stacking, and the core of this process is to generate base models that have diversity. There are three influencing factors and parameters that could lead to such diversity: architectures, training data, and sliding window size.

#### 3.2.1 Architectures of Base Models

The driving idea behind stacking is to train different base models with the hope that they can learn different features of data. The architecture of a model exactly decides how this model works. In this research, we designed six different architecture for base models. Three of them are based on CNN while the other three are based on LSTM. The reason of choosing these two variants is that CNN and LSTM process data from different perspectives. For CNN, it uses convolution to extract spatial features from data. For LSTM, it analyzes temporal information with the help of three special gates in LSTM units, which are explained in Chapter 2. The detail of these six base models is shown in Figure 11-16.

The three CNN base models in Figure 11-13 are different in their convolution layers and fully-connected layers. CNN-1 has four convolution layers and two fully-connected layers. Each convolution layer is followed by a max pooling layer. CNN-2 consists of eight convolution layers instead of four. Finally, CNN-3 has four convolution layers. But it has two more fully-connected layers than CNN-1 and CNN-2. It should be noted that, as mentioned in Chapter 2, in CNN, convolution layers plays the role of feature extraction. On the other hand, fully-connected layers do the job of classification. As a result, CNN-2 put more efforts into feature extraction

than CNN-1 since CNN-2 has more convolution layers. On the other hand, CNN-3 focus more on classification due to its two more fully-connected layers than CNN-1.

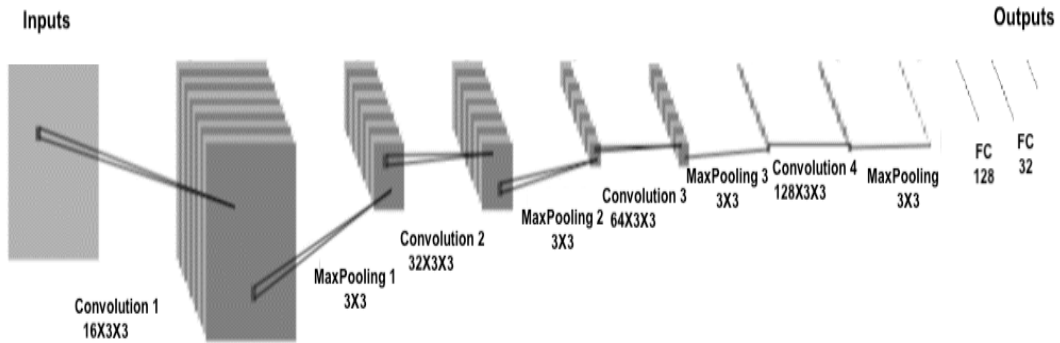


Figure 11 The architecture of base model CNN-1 with four convolution and two fully-connected layers (FC).

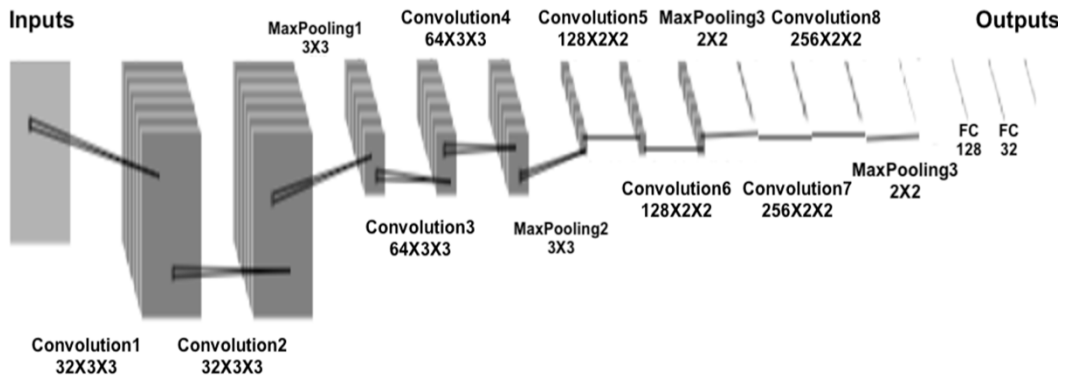


Figure 12 The architecture of base model CNN-2 with eight convolution and two fully-connected layers

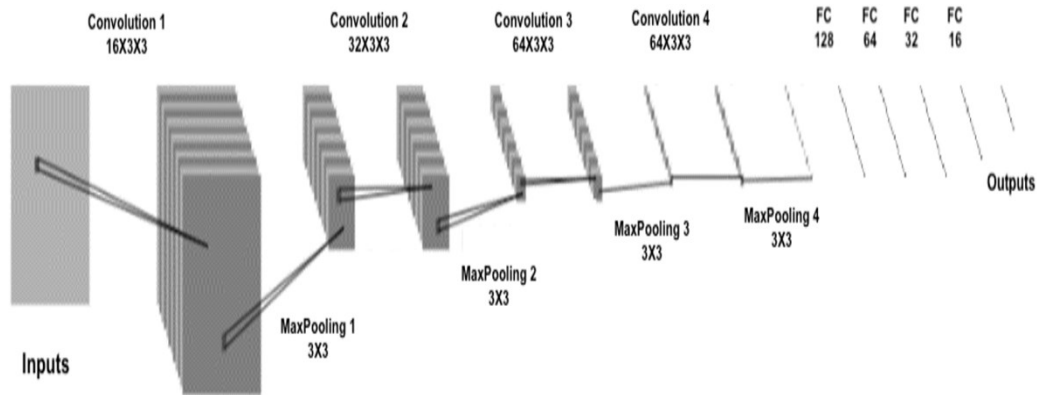


Figure 13 The architecture of base model CNN-3 with four convolution and four fully-connected layers

Similar to the three CNN base models, the three LSTM base models in Figure 14-16 are different in their LSTM layers and fully-connected layers. For LSTM, its LSTM layers extract features and pass them to fully-connected layers that fulfill the classification. Shown as Figure 14, LSTM-1 consists of one LSTM layer with 128 LSTM units and one fully-connected layer. LSTM-2, in Figure 15, also has one fully-connected layer but one more LSTM layer to enhance its ability of feature extraction. In more detail, the first LSTM layer of LSTM-2 has 256 units, and the second LSTM layer has 128 units. Lastly, LSTM-3 consists of one LSTM layer with 128 LSTM units and two fully-connected layers. Compared with LSTM-1, LSTM-3 pays more resources to classification.

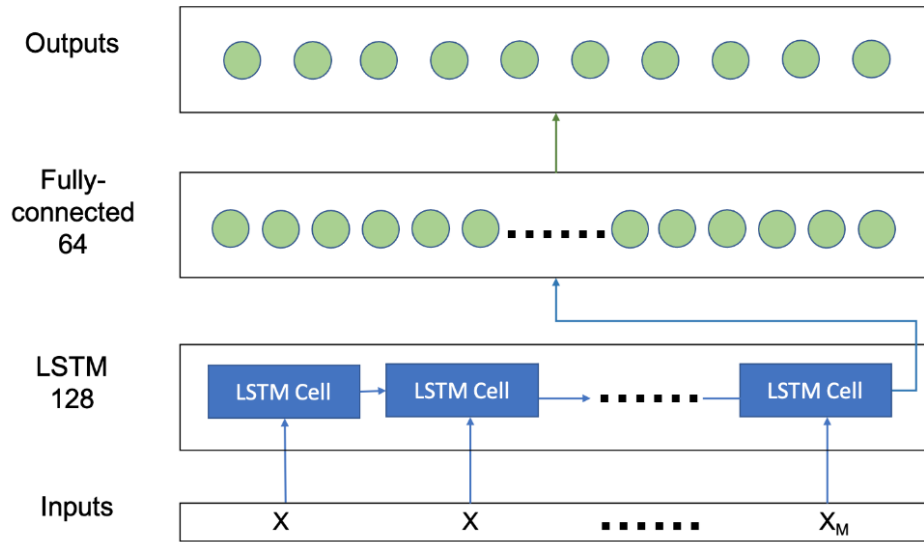


Figure 14 The architecture of base model LSTM-1 with one LSTM layer and two fully-connected layer

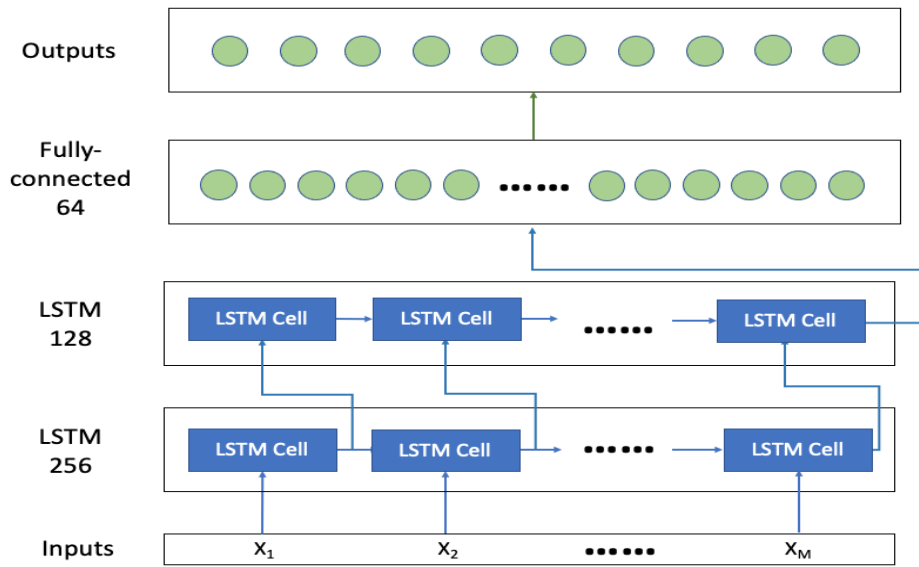


Figure 15 The architecture of base model LSTM-2 with two LSTM layer and two fully-connected layer

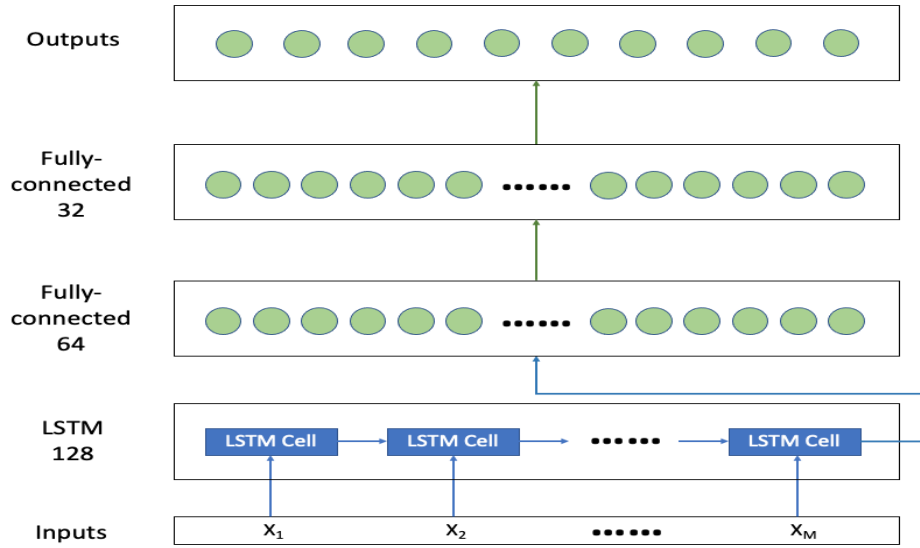


Figure 16 The architecture of base model LSTM-3 with one LSTM layer and two fully-connected layer

By having such six different architectures, base models can make predictions from different perspectives to maintain the diversity of base models.

### 3.2.2 Training Data of Base Models

Training data is crucial in deep learning, which could directly influence the performance of the model regardless of the architecture of the model [43]. With the motivation to achieve the diversity from the perspective of training data, [44] applied k-fold training set to generate k folds and used these k different folds to further feed base models, which we will adapt in this work. To be more particular, we first divide the whole training data into k folds evenly. Next, we build k training subsets. Each training subset consists of k-1 folds. During training, each base model will randomly select one training subsets from them.

### 3.2.3 Sliding Window Size of Base Models

Sliding window size is the number of sensor events that a particular input contains. It identifies how much and what kind of information the model could acquire and utilize. Naturally that the sliding window size should be as large as possible since the more information about previous activities the model can recall, the better the performance should be. However, some of the

previous activities could be redundant [40]. For instance, going to the toilet one morning far in the past will not impact the current activity of cooking dinner. Hence, the sliding window size should be a reasonable value.

In this work, we consider five different sliding window sizes: 50, 100, 150, 200, 250, 300. The reason for not considering higher values is that higher values will require computation more than what our devices used to run the model in this research can afford.

### 3.3 Meta Model Based on LSTM

The objective of a meta model is to decide how the base model outputs are utilized by combining the results from previous base models. Looking at this from another angle, the meta model acts as a classifier with the outputs of the base models as the input features. The core idea of the meta model is to learn whether base models have learnt the training data correctly. For example, if a base model incorrectly learned certain patterns of features extracted from the training data, and hence consistently misclassifies samples observing those patterns, then the meta model may be capable of learning this behavior and correct the predictions.

Since the meta model takes output from base models as input, that output should be pre-processed in advance. Shown in Table 4, all output from base models is merged as one matrix. In this matrix, each row is the prediction made by a base model. Assume there are  $m$  base models selected. Then the input can be expressed as:

Table 4 The input of the meta model

	Activity 1	.....	Activity n
$B_1$	$X_{1,1}$	.....	$X_{1,n}$
.....	.....	.....	.....
$B_m$	$X_{m,1}$	.....	$X_{m,n}$

Where  $n$  is the number of types of activities,  $X_{m,n}$  is the probability that the activity is predicted as Activity  $n$  by  $m^{\text{th}}$  selected model, and  $B_m$ , is the  $m^{\text{th}}$  base model.



This matrix consisting of predictions made by the base models is put into a meta model based on LSTM. A typical LSTM model consists of LSTM layers and fully-connected layers. The LSTM layer extract features from input data, and these features are further processed by the following fully-connected layers to make predictions. The meta model used in this research is shown as Figure 17. This meta model has one LSTM layers and one fully-connected layer. The first LSTM layer consists of 64 LSTM units. The following fully-connected layer has 32 neurons which process the features passed from previous LSTM layer to make the predictions. By having such arhchitecture, the inputted matrix's shape is transformed from  $M \times N$  to 64, then to 32, finally to the size of outputs. It should be noticed that if the number of LSTM units after the input layer is too bigger or too close or too smaller than the number of base models  $M$ , the LSTM model will be either underfitting or overfitting. Overfitting refers to the situation where a deep learning model has too much neurons to achieve good performance. On the other hand, underfitting is the phenomenon where the deep learning model is trained not enough because the model itself has fewer neurons than expected to extract features. In our case, the largest number of base models will be 30. Therefore we select 64 LSTM units. For the following fully-connected layer, it connects the LSTM layer and the output layer. It should have neurons between 64 and the size of the output. In our case, the size of the output is the number of activties. Look back to Table 2, the lagerst number of activities we have among six datasets is 31. Therefore, we choose 32 neurons in the fully-connected layer.

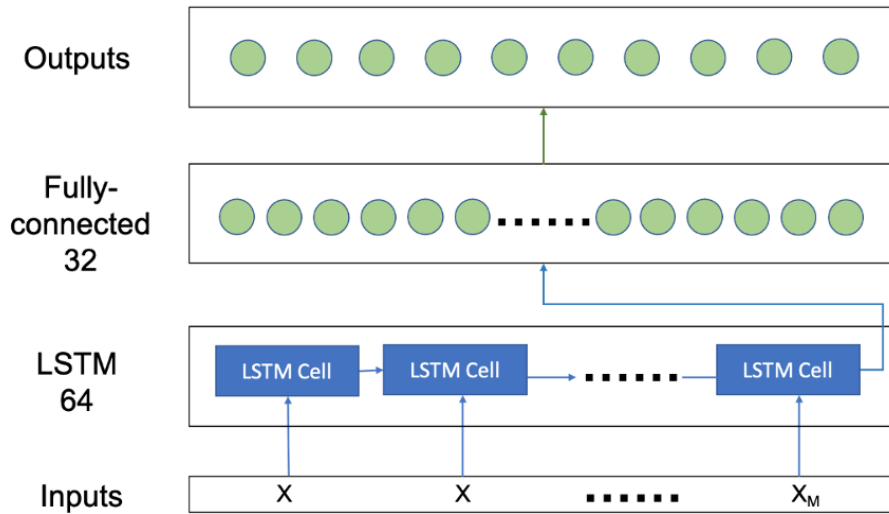


Figure 17 The architecture of the meta model

### 3.4 Pre-test

In order to test whether the diversity of architectures, training data, and sliding window size influence on the performance, we conducted a pre-test. In this pre-test, we only use the data from the first month. Also, we split the data into three sets: set 1, set 2, and set 3. These three sets are used as training data, validation data, and test data set, respectively.

#### 3.4.1 Impact of Diversity of Architectures

To test the influence caused by the diversity of architectures, we first trained different base models with different parameters. Then we selected the base model with highest accuracy on the test data from set 3 as the benchmark and build another two stacking models: Model 1 and Model 2. Model 1 and 2 have the same meta model instead of same base models. For Model 1, it has six base models with the same architecture of benchmark's. However, Model 2 has six different base models with different six different architectures in Section 3.2.1. Therefore, benchmark, Model 1, and Model 2 represent non-stacking model, stacking model without the diversity of architectures, and stacking model having the diversity architectures. Figure 18 shows their performances on the datasets, measured by prediction's accuracy. We also compared these

three models by applying the Friedman test with assuming they are similar. In brief, the Friedman test is a statistical test that is suitable to detect difference in machine learning algorithms across multiple test attempts [45]. By assuming there is no significant difference, the p value is 0.0111, which is smaller 0.05. This meant there was a significant difference in this group. In order to explore the cause of this difference, we conducted the Nemenyi test. Nemenyi test intended to find which groups of comparisons cause the significant difference when such significant difference is proved by the Friedman test. Figure 19 shows the result of the Nemenyi test. The result shows that the p value of the group of benchmark model and Model 2 and the group of Model 1 and Model 2 are both smaller than 0.05, which indicates Model 2, which consisted of various models with different architectures, is significantly better than the other two models. In other words, the diversity of architectures does have positive influence on the performance.

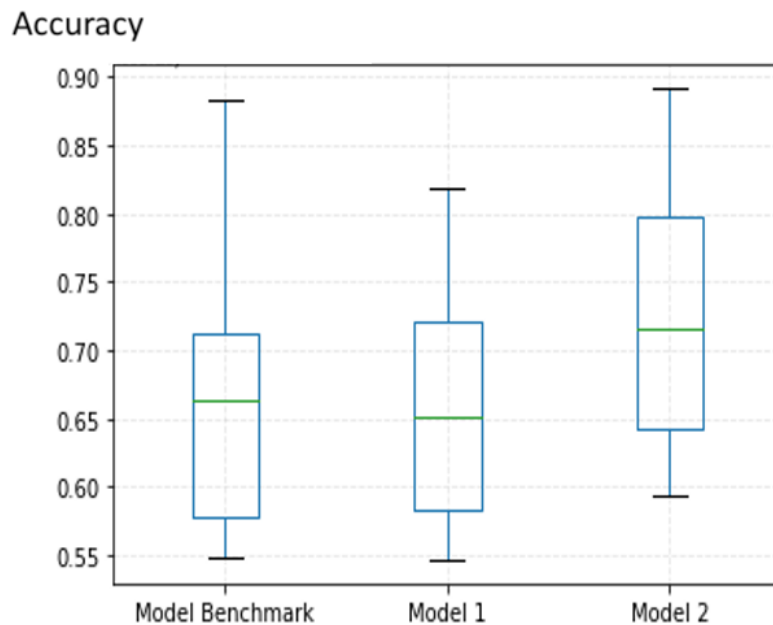


Figure 18 Box plot comparison of the accuracy of Model Benchmark, Model 1, and Model 2

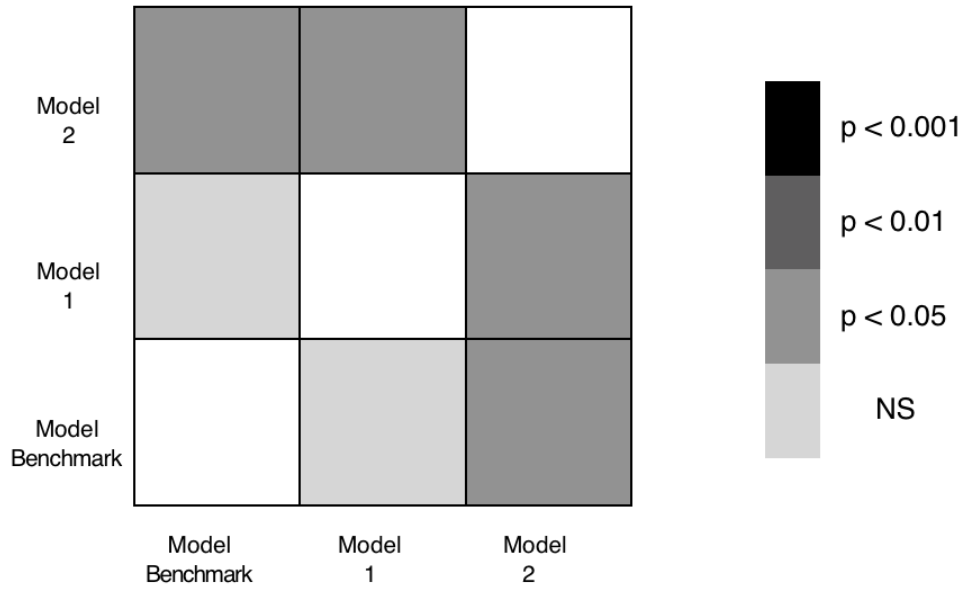


Figure 19 The result of Nemenyi test for group of Model Benchmark, Model 1, and Model 2

### 3.4.2 Impact of Diversity of Training Data

Like the pre-experiment in the last section, there also were three models trained to test the influence caused by the diversity of architectures: Model Benchmark, Model 3 whose base models were trained by same training data, and Model 4 whose base models were trained by different subsets of training data. Figure 20 shows their accuracy. We further conducted the Friedman test with assuming they are similar, and the p value is 0.0111, which is smaller than 0.05. Therefore, we could conclude that there were significant differences between the three models. To figure out what caused it, we did the Nemenyi test. The result of the Nemenyi test is shown in Figure 21. The result shows that the p value of the group of Model Benchmark and Model 4 and the group of Model 3 and Model 4 are both smaller than 0.05, which indicates Model 4, which consisted of various models trained by different data, is significantly better than the other two models. In other words, the diversity of training data does have positive influence on the performance of HAR using ambient sensors.

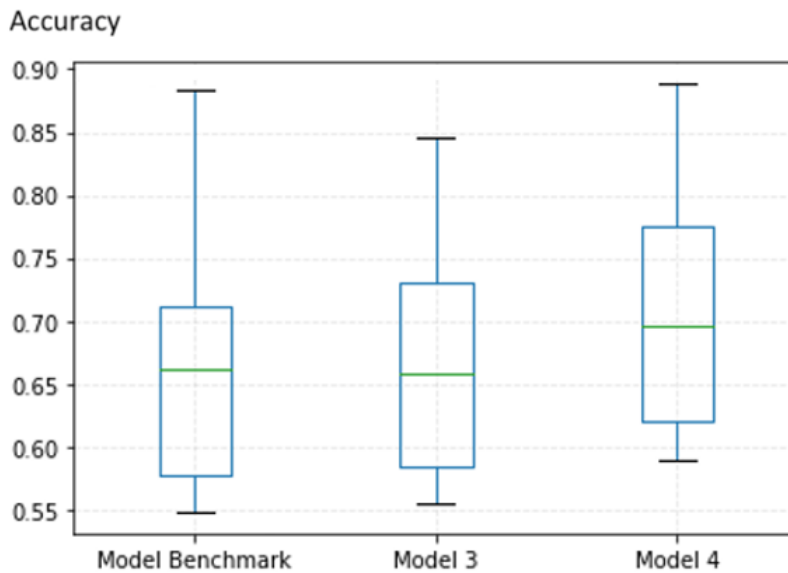


Figure 20 Box plot comparison of the accuracy of Model Benchmark, Model 3, and Model 4

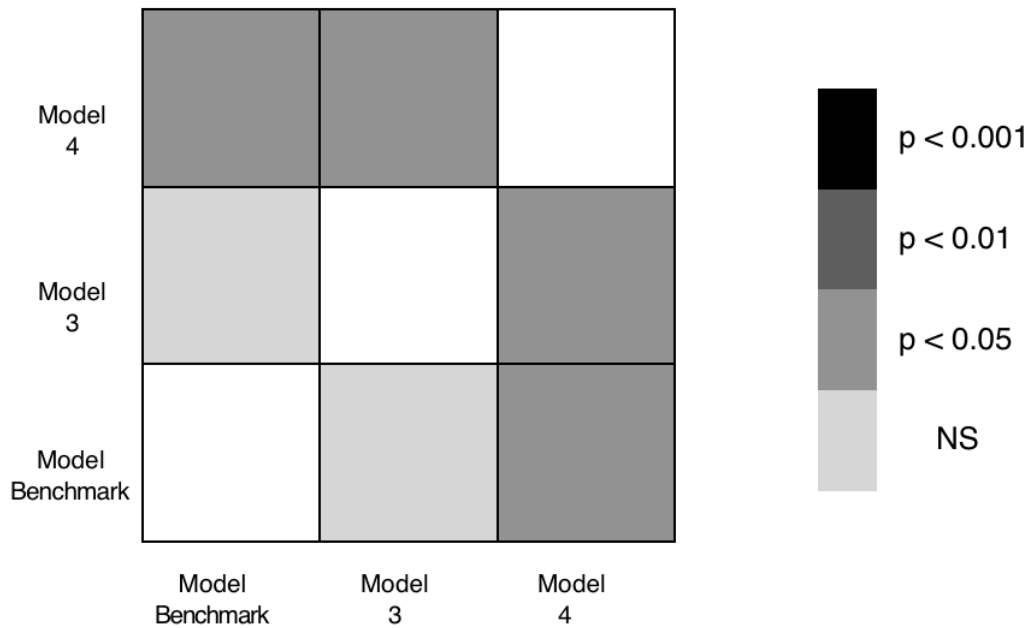


Figure 21 The result of Nemenyi test for group of Model Benchmark, Model 3, and Model 4

### 3.4.3 Impact of Diversity of Sliding Window Size

Three models were used to test the influence caused by the diversity of sliding window size.

Besides the Model Benchmark in previous two sections, there were also another two models:

Model 5 and Model 6. Model 5 was trained with base model of same sliding window size. However, Model 6 consisted of base models with different sliding window sizes. Figure 22 shows their performances measure in accuracy. The result of the Friedman test shows that models in groups were significantly different, given that the p value was 0.0057, smaller than 0.05. A Nemenyi test was done to explore the reason behind the difference. As shown in Figure 23, the p value of the group of Model 6 and Model Benchmark is smaller than 0.05. This demonstrated that stacking base models with various sliding window sizes do help Model 6 significantly outperform Model Benchmark, which means that maintaining the diversity of sliding window sizes indeed is helpful.

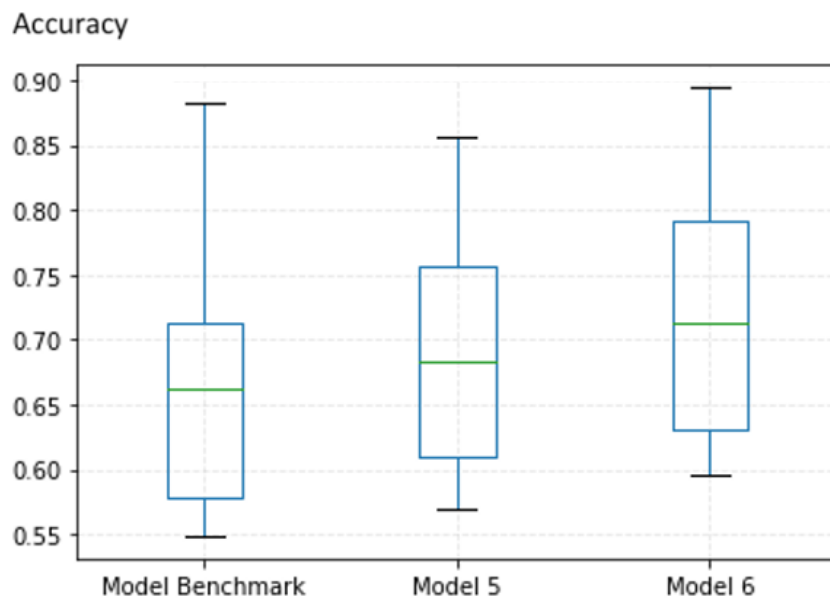


Figure 22 Box plot comparison of the accuracy of Model Benchmark, Model 5, and Model 6

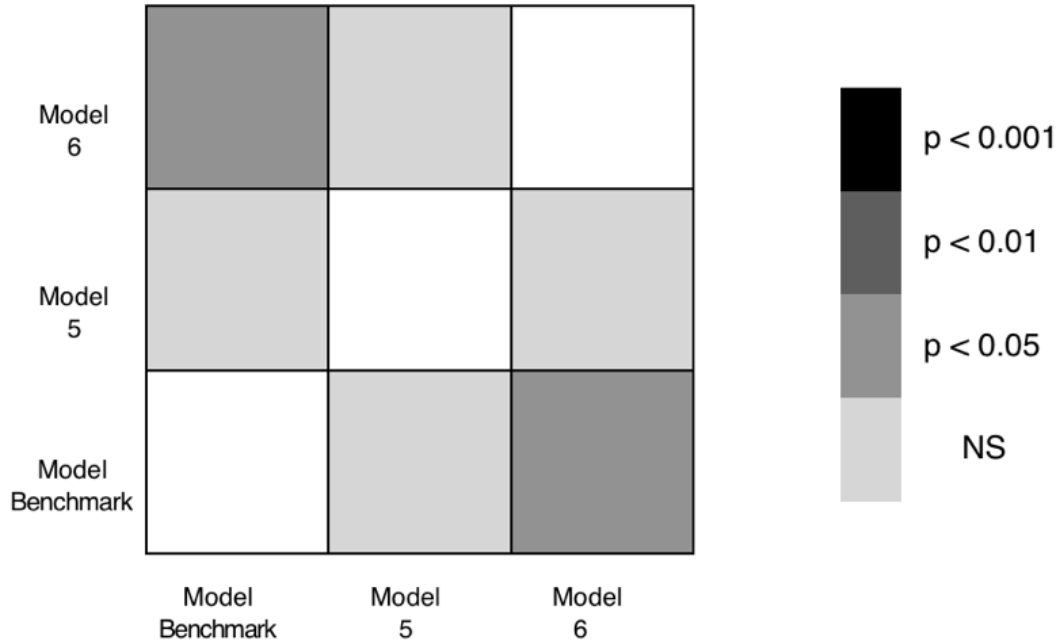


Figure 23 The result of Nemenyi test for group of Model Benchmark, Model 5, and Model 6

#### 3.4.4 Pre-test Conclusion

In Section 3.4, we conducted a pre-test. This pre-test tested the performances of three groups of models to explore the influence of three factors, including the architecture, training data, the sliding window size, on the performance of stacking model. To test the difference between results, we also conducted both Friedman test and Nemenyi test. The results of these two statistical tests demonstrated that varying three factors mentioned above did improve the performance. Therefore, we decided to build base models with varying their architectures, training data, and sliding window sizes.

### 3.5 Proposed Deep Learning-based Stacking Method for HAR Based on Ambient Sensors

---

Algorithm 1: Deep learning-based stacking for HAR based on ambient sensors

---

Input: Training Data

Output: Stacking model based on deep learning

Split the data into Training Data, Validation Data, and Test Data.

Split Training data into K folds.

for  $i=1$  to M do

    Randomly select a fold k. Build a training set consisting of data from K-1 folds except the fold k.

    Randomly select an architecture from the 6 architectures in Section 3.2.1.

    Randomly select a sliding window size from 6 sliding window sizes in Section 3.2.3.

    Construct a base model-i with the selected architecture and sliding window size,

    Train a base model-i with training data from the training set built above.

    Generate Output-i by inputting data from Training Data and Validation Data.

    Generate Prediction-i by inputting data from Test Data.

end

Construct a set of training data for the meta model by merging M base models' outputs.

Construct a set of test data for the meta model by merging output from Prediction 1 to

Prediction M.

Train the meta model based on deep learning with training data for meta model and test data for meta model.

Output the stacking model.

---



Alg. 1 summarises the method we developed. First, the dataset is split into three folds: Training Data, Validation Data, and Test Data. Then Training data is broken down into  $K$  folds to train  $M$  base models with different architectures and sliding window sizes. After finishing training base models, we input Training Data and Validation Data into base models to generate their prediction. We further merge the prediction into another dataset as the training data for the meta model. At last, the meta model is fed by the training data for meta data trained to get the final stacking model.

### 3.6 Ambient Sensor Generative Adversarial Network

Besides developing algorithms for recognizing activities, insufficient data is another problem in HAR using ambient sensors. The proposed ambient sensor generative adversarial network (ASGAN) aims to generate synthetic ambient sensor data to enlarge the training dataset and further improve the performance of HAR models.

A typical GAN aims to learn the data distribution from a set of real data to further generate synthetic data drawn from the learnt distribution. GAN has two components: a generator and a discriminator. For the generator, it takes noise as input and transform them to synthetic data. For the discriminator, the goal is to distinguish real data and synthetic data.

Our proposed ASGAN has the architecture shown as Figure 24. There is a generator and a discriminator. We denote these two components as  $G$  and  $D$  respectively. The generator plays the role of generating synthetic data while the discriminator tells whether received data is from real data, labeled as real, or synthetic data, labeled as fake. Given that ambient sensor data streaming for HAR has strong temporal features and LSTM's strong ability of dealing with temporal data, we build both generator and discriminator based on LSTMs.

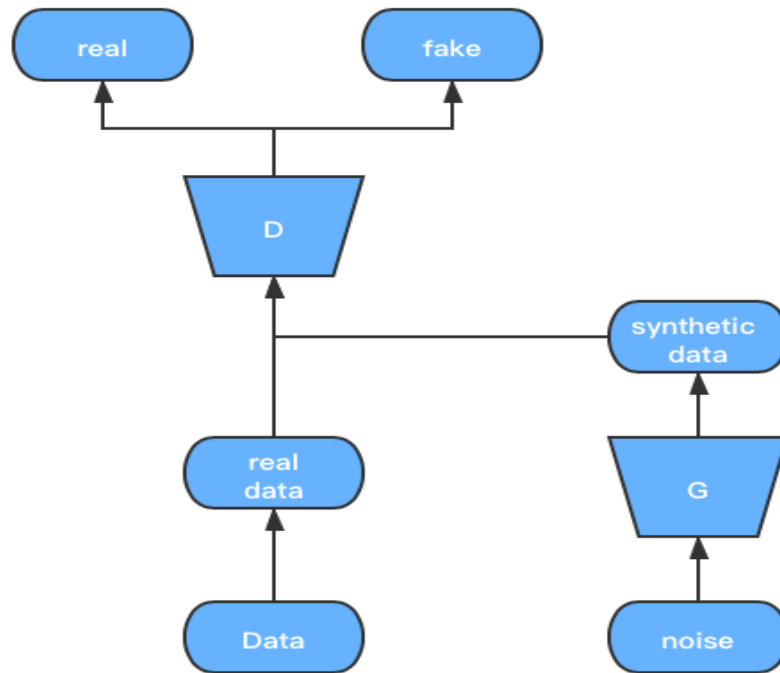


Figure 24 Proposed ASGAN's architectures. This model consists of two components: generator G and discriminator D.

The architecture of generator is demonstrated as Figure 25. There are three LSTM layers. They have 128, 32, 1 LSTM units respectively. To discuss the motivation behind this architecture, let's look back to Section 3.2.3. In that section, we mentioned that we considered six sliding window sizes, ranging from 50 to 300. To make sure that our synthetic data can be used by all base models regardless of the sliding window size they take, we set the synthetic data's value of sliding window size as 300. In this way, such synthetic data are trainable for all base models. For instance, base model with sliding window size of 50 can use the synthetic data with sliding window size of 300 by only inputting the last 50 rows of the synthetic data. The first two LSTM layers capture the temporal characteristic from input. The last LSTM layer then uses such characteristic to generate sequences of sensor data. Now look back to the architecture of generator. The output of the first LSTM layer, with 128 LSTM units, is a  $300 \times 128$  matrix,

given that a LSTM layer's output is a matrix whose number of rows is the input size and number of columns is the number of LSTM units. After the second LSTM layer with 32 units, the output becomes  $300 \times 32$ . Finally, the last LSTM later with only 1 LSTM unit outputs 300 sensory data containing the sequential information of ambient sensor's activation. The number of LSTM units in each LSTM layer is getting decreased in order to reshape the output from  $300 \times 128$  to  $300 \times 1$ . Also, there are 128 units in the first LSTM layer because more units will take more computation resources than we can accept. Figure 26 demonstrated the architecture of discriminator, which has four LSTM layers. The first three layers extract features from input. Then the last layer labels the real data as real and the synthetic as fake.

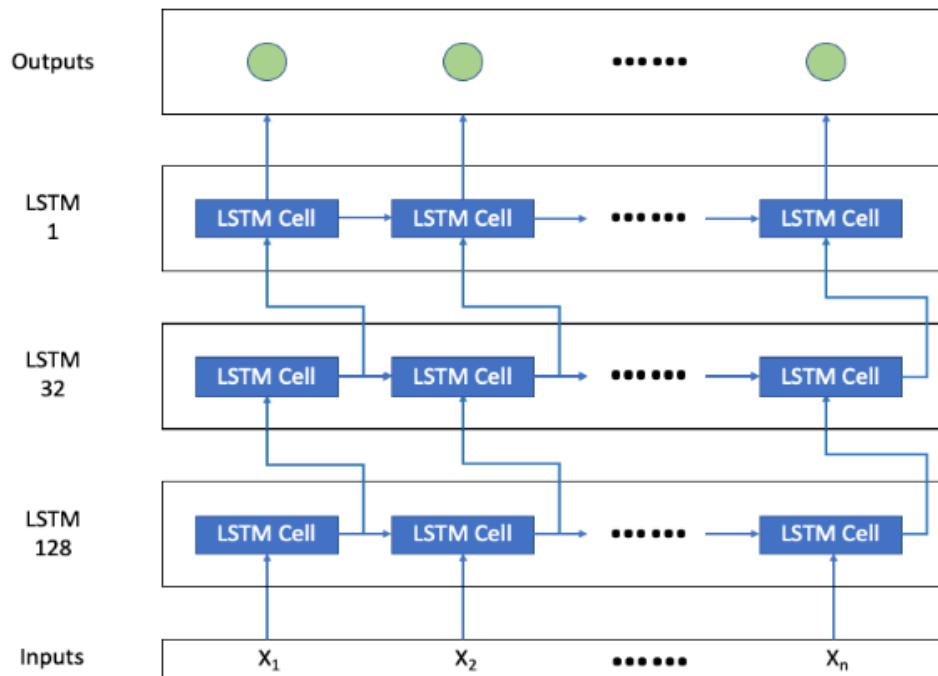


Figure 25 Generator of proposed ASGAN

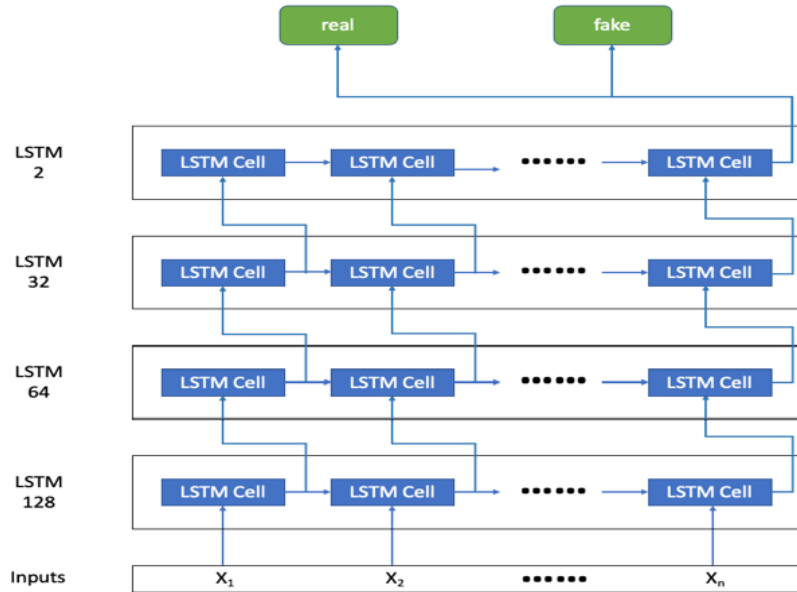


Figure 26 Discriminator of proposed ASGAN

---

Algorithm 2: Training Process algorithm for ASGAN

---

Input: real data

Output: synthetic data

for  $i=1$  to  $N$  do

Generate 32 noise from standard normal distribution.

Input noise above into generator  $G$  to obtain synthetic data.

Draw 32 samples from real training data.

Train discriminator  $D$  by feeding above synthetic data and real data samples.

Generate another 32 noise from standard normal distribution and feed them to generator with minimizing  $V(G)$  in Equation 7.

end

Use  $G$  to generate synthetic data

---

Algorithm 2 summarizes the training process for the proposed ASGAN. First, we generate 32 noise from standard normal distribution. The choice of 32 and normal standard distribution is based on [46], G will use such 32 noise as input to generate k batches of synthetic data. Afterwards, 32 samples from real training data will be randomly selected to train D along with 32 data generated previously. During training, samples from real data will be labeled as real. On the other hand, samples from synthetic data will be labeled as fake. The goal of D is to distinguish between real and fake. After training D, another 32 noise will be drawn from standard normal distribution. Next, use the noise to train G with minimizing the value  $V(G)$  in Equation 7, where  $G(z)$  is the synthetic data generated by G,  $D(G(z))$  is the probability that  $G(z)$  being predicted as real by D, then  $1 - D(G(z))$  is the probability that  $G(z)$  being predicted as fake by D. Here,  $V(G)$  is the expectation of synthetic data being predicted as fake by D. By minimizing  $V(G)$ , G improves its ability of fooling D. In other words, G is trying to generate synthetic data whose probability being predicted as fake by D is as low as possible.

$$V(G) = E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (7)$$

After repeating above steps for N iterations, we use this G to generated synthetic data to enlarge the training dataset with the goal to improve the performance of HAR models.

In this chapter, we first proposed a deep learning-based stacking neural network for HAR using ambient sensors. This stacking model consists of two parts. The first part is the base models with different architectures, training data, and sliding window sizes. The second part is the meta model. In this stacking model, base models first make predictions based on inputs. Then the meta model will make the final prediction by using above base models' predictions. We also proposed an ASGAN. ASGAN is a generative adversarial network for ambient sensor data. Both generator

and discriminator of this ASGAN are based on LSTMs. An algorithm of how to train ASGAN is also presented.

To test these two proposed methods, the next chapter will conduct two experiments. The first experiment is to test the deep learning-based stacking neural network for HAR using ambient sensors. The second to test whether synthetic data generated by ASGAN can improve the performance of HAR using ambient sensors.

## Chapter 4: Experiments and results

In this chapter, our proposed methods are applied on the six datasets from CASAS dataset. First, we compared the proposed deep learning-based stacking method for HAR using ambient sensors with the best among other works in the literature. Then we conducted another experiment to test the ASGAN's influence on the performance of HAR. All the models in experiments were executed using a single NVIDIA GeForce 750Ti GPU with 2GB memory. The programming was conducted using TensorFlow under Python environment.

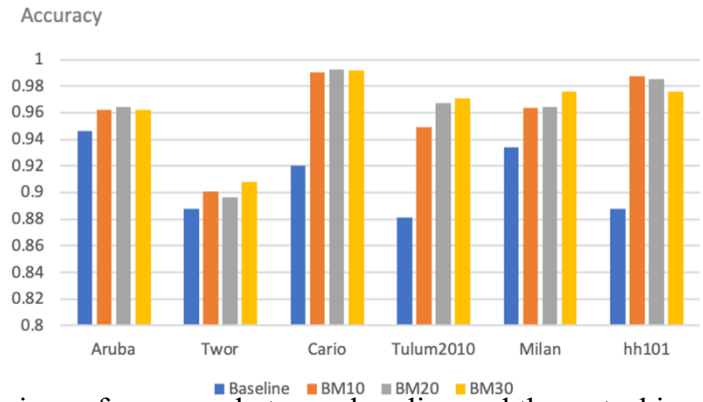
### 4.1 Deep Learning-Based Stacking Method for HAR Using Ambient Sensors

We trained our proposed deep learning-based stacking method for HAR using ambient sensors on six datasets from CASAS.

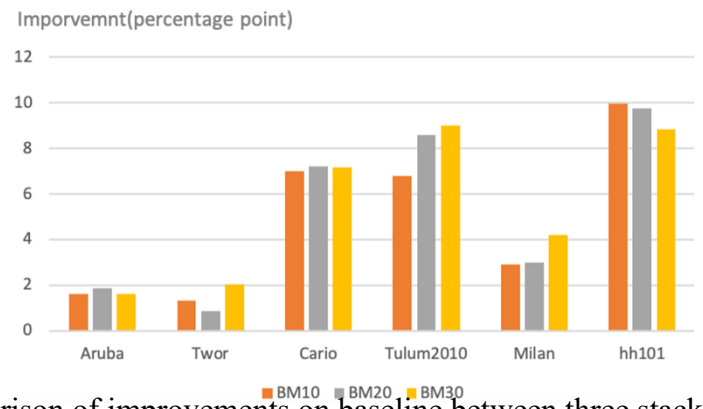
For each dataset, we split the data into training data for base models, validation data for base models, training data for the meta model, validation data for the meta model, and the test data for the meta model. Training data for base models and training data for the meta model are used to train base models and the meta model respectively. Validation data for base models and validation data for the meta model play the role of early stopping, which is the phenomenon where a model's training will be stopped if its performance on the validation data will not be improved anymore. Test data for the meta model are employed to evaluate the meta model's performance.

Next, we trained 3 different models by using proposed deep learning-based stacking method for HAR using ambient sensors. These 3 models are different in the number of base models. The first model is named as Model BM10. It has the 10 base models (BM). The second model, Model BM20, has 20 base models. And the third model, Model BM30, has 30 base models. Also, we considered the best [42, 47-49] among other works found by the author in the literature as the

baseline. The results are evaluated by the human activity recognition accuracy. This accuracy is defined by Equation 8, where TP refers to true positives, TN represents true negatives, FP refers to false positives, and FN represents false negatives.



(a) Comparison of accuracy between baseline and three stacking models



(b) Comparison of improvements on baseline between three stacking models

Figure 27 Results of experiments on three stacking model compared with baseline

Figure 27 shows the result. It can be seen all three models' human activity recognition accuracy outperformed the baseline over every dataset by minimum 0.87 percentage points and maximum 9.96 percentage points. Also, we conducted significance tests to investigate whether there is a significance statistical differences between our proposed method and other works. The significance test is conducted in pairs. Although t-test is the most commonly used method for significance test, it is not suitable when it comes to test different algorithms or models' performance over different datasets due to the three drawbacks of t-test method [45]. Firstly, the t-test only makes sense when the differences over the data sets are commensurate. In other



words, the expected accuracy of a model’s predictions on every dataset should be the same. However, such commensurability is not achievable because, in real world, each dataset has its own features such as activity distribution, sensors layout, etc. Secondly, t-test requires that tested data should be normally distributed. Considering that the nature of our dataset is about human activities, which is not guaranteed to observe any distribution, the requirement for normal distribution cannot be guaranteed either. Thirdly, t-test needs 30 datasets at least, which cannot be satisfied since we only have six datasets in our work. [45] suggests that Wilcoxon signed-ranks test would be more proper when it comes to machine learning because Wilcoxon signed-ranks test does not have above three drawbacks. It can be applied to datasets without the requirement for commensurability, normal distribution, or sizes (more than 30 datasets). Hence, we decided to use the Wilcoxon signed-ranks test.

$$\text{Human activity accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (8)$$

$$R^+ = \sum_{d_i>0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i=0} \text{rank}(d_i) \quad (9)$$

$$R^- = \sum_{d_i<0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i=0} \text{rank}(d_i) \quad (10)$$

Wilcoxon signed-ranks test ranks the difference in performance (activity recognition accuracy in our case) of two models for each dataset, ignoring the signs, and compares the ranks for positive differences and negative differences. Let  $R^+$  be the sum of ranks for the datasets on which the second algorithm outperformed the first, and  $R^-$  be the sum of ranks for the opposite. These two sums can be obtained by Equation 9 and Equation 10. Where  $d_i$  is the difference between the performance of the two compared models on i-th dataset,  $\text{rank}$  is the rank function. Detail on how to calculate this rank will be provided in an example of BM10: the result of comparison between BM10 and the baseline is shown in Table 5. The difference between baseline’s and BM10’s accuracy on Aruba is 0.0159. Given 0.0159 is positive, BM10’s accuracy on Aruba has

a sign of +1 (-1 if the difference is negative, 0 if zero). Also, its index ranked by absolute difference is 2, given 0.0159 is the second smallest absolute difference in Table 5. Therefore, the Wilcoxon signed-ranked test is  $(+1) * 2 = +2$ . Similarly, we can get the Wilcoxon signed-ranks on all six datasets, shown in Table 5. By using Equation 9 and Equation 10, we can get  $R^+ = 21$  ( $+2 + 1 + 5 + 4 + 3 + 6$ ) and  $R^- = 0$ . Hence, let  $T = \min(R^+, R^-) = 0$ . Looking up the table of exact critical values for  $T$  with  $\alpha = 0.05$  and  $N = 6$  [45], the conclusion can be made that if  $T = 0$ , the null hypothesis can be rejected. Therefore, it can be concluded that BM10 is significantly better than the baseline. The results of the rest models are represented in Table 6. It can be seen that our three models are all significantly better than the baseline model.

Table 5 Wilcoxon signed-ranks test for BM10

Dataset	Baseline's accuracy	BM10's accuracy	Difference	Absolute Difference	Sign	Index Ranked by Absolute Difference	Wilcoxon Signed-Rank
Aruba	0.9460	0.9619	+0.0159	0.0159	+1	2	+2
Twor	0.8875	0.9006	+0.0131	0.0131	+1	1	+1
Cario	0.9200	0.9901	+0.0701	0.0701	+1	5	+5
Tulum2010	0.8810	0.9490	+0.0680	0.0680	+1	4	+4
Milan	0.9342	0.9633	0.0291	0.0291	+1	3	+3
hh101	0.8878	0.9874	0.0996	0.0996	+1	6	+6

Table 6 Wilcoxon signed-ranks test results regarding proposed stacking method

Comparison	R+	R-	T	Significance
BM10 and baseline	21	0	0	Significant
BM20 and baseline	21	0	0	Significant
BM30 and baseline	21	0	0	Significant

## 4.2 Ambient Sensor Generative Adversarial Network

To test whether our proposed ASGAN can help improve the HAR performance by enlarging the training dataset, we trained 3 different models: Model BM10-ASGAN, Model BM20-ASGAN, Model BM30-ASGAN. These three models all share the same base models of BM10, BM20, and BM30. However, they are all trained by training data sets enlarged by ASGAN instead of training data sets consisting of mere real data. In detail, both training data for base models and training data for meta model are enlarged by ASGAN. The ratio of training data from real data and synthetic data generated by ASGAN is 1:1.

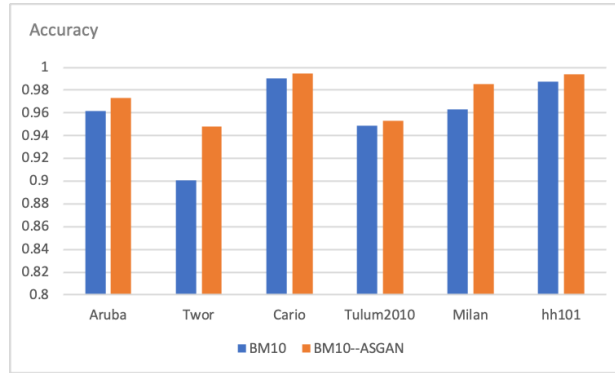
Table 7 Improvement (percentage points) made by ASGAN

Dataset	BM10—ASGAN and BM10	BM20--ASGAN and BM20	BM30--ASGAN and BM30
Aruba	1.15	1.68	0.44
Twor	4.77	1.05	2.84
Cario	0.45	0.42	0.2
Tulum2010	0.44	0.87	1.56
Milan	2.21	1.06	0.96
hh101	0.66	2.1	0.15
Average	1.61	1.20	1.03

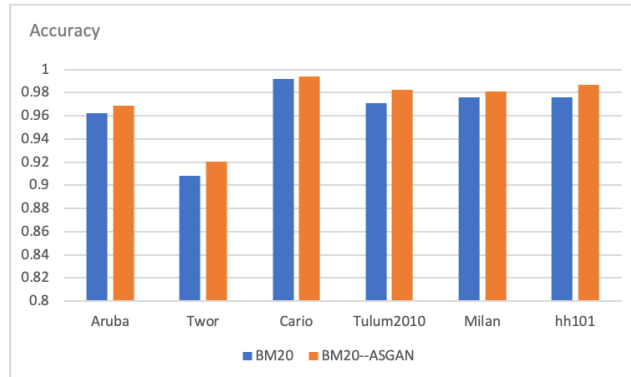
Table 8 Wilcoxon signed-ranks test results of ASGAN

Comparison	R+	R-	T	Significance
BM10 and BM10-ASGAN	21	0	0	Significant

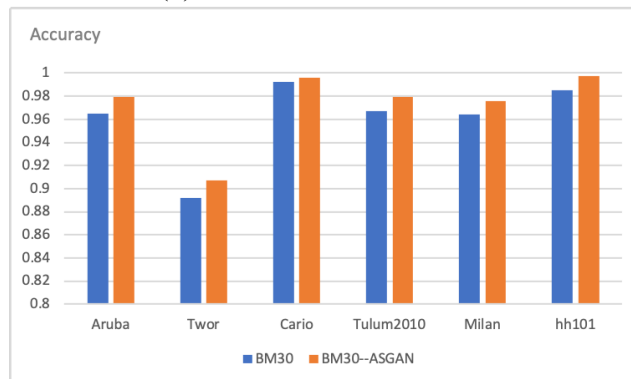
BM20 and BM20-ASGAN	21	0	0	Significant
BM30 and BM30-ASGAN	21	0	0	Significant



(a) BM10 and BM10-ASGAN



(b) BM20 and BM20-ASGAN



(c) BM30 and BM30-ASGAN

Figure 28 Comparison of accuracy between three pure data driven models and three mixed data models whose training data are enlarged by ASGAN

Figure 28 and Table 7 presents the results of comparison of accuracy and the improvements made by models trained by data enlarged by ASGAN. It can be seen all three models trained by

enlarged data outperforms models trained by mere real data. Similar to Section 5.1, we also conducted Wilcoxon signed-ranks test in pairs. The results of Wilcoxon signed-ranks tests are shown in Table 8. It can be seen that models augmented by ASGAN are all significantly better than their corresponding models that are only trained by real data. Also, it can be seen that the improvements achieved by ASGAN can be guaranteed regardless of the dataset or the number of base models (the minimum is 0.15 percentage points while the maximum is 4.77 percentage points). This proves that ASGAN can help HAR model accomplish a stable improvement.

## Chapter 5: Conclusions and Future research

Home healthcare is an increasingly growing industry stimulated by the phenomenon of population ageing. Among various home healthcare services, smart home for elderly care is the fastest growing one. As the key aspect of the smart home for elderly care, HAR has gained more and more attentions from researchers. Compared with HAR using cameras and wearable sensors, HAR using ambient sensors is getting more popular because it does not cause resident's concern for privacy like camera-based HAR or require to be worn all the time like wearable sensor-based HAR. One of the primal concerns of HAR using ambient sensors is to construct a model that can recognize activities with high accuracy. Besides, the problem of insufficient data is another concern.

This thesis aims to tackle these two problems. Firstly, we proposed a deep learning-based stacking method for HAR using ambient sensors. This method combined deep learning and stacking to construct a model that can recognize activities with high accuracy. Secondly, we constructed a GAN-based model, ASGAN, to tackle the problem of insufficient data. This ASGAN enlarged the training dataset by generating synthetic data.

To test proposed methods, we conducted two sets of experiments. The first set compared baseline, the best among other works we could find, and the proposed deep learning-based stacking method on several datasets. The second experiment tested the influence of synthetic data on the stacking model's performances. Wilcoxon signed-ranks tests are also conducted to prove the results' statistical significance.

The first experiment's results show that the proposed deep learning-based stacking method for HAR using ambient sensors statistically improved the activity recognition accuracy by minimum of 0.87 percentage points and maximum of 9.96 percentage points, and the average is 5.21

percentage points. This improvement is achievable regardless the number of base models, given three stacking models (BM10, BM20, BM30) with different number of base models all outperform the baseline.

The second experiment's results show that models trained by training datasets enlarged by ASGAN statistically outperform models trained only by real data. Particularly, synthetic data generated by ASGAN help models achieved improvements on the activity recognition accuracy by minimum of 0.15 percentage points, maximum of 4.77 percentage points, and average of 1.28 percentage points. Such improvements prove that proposed ASGAN indeed has positive influence on the performance of HAR model. What's more, this improvement achieved by ASGAN is stable, given that this improvement was accomplished on all six used datasets regardless of the number of base models.

In summary, the contributions of this thesis to the area of HAR using ambient sensors are:

- We proposed a deep learning-based stacking neural network for HAR using ambient sensors. This deep learning-based stacking neural network's human activity recognition accuracy outperformed the best model among other works we could find in the area of HAR using ambient sensors.
- We designed an ambient sensor generative adversarial network (ASGAN) to tackle the problem of insufficient faced by HAR using ambient sensors. This ASGAN can lead to stable improvements on the human activity recognition accuracy by enlarging the training data sets with generated synthetic data.

There are still some future improvements that could be done on this research. Firstly, this research only focused on recognizing daily activities. However, there are some critical activities (e.g. fall, coma) that rarely happen and are not considered by most HAR related researches. Such

activities needed to gain more attention because residents could be in dangerous health condition once they happened. Therefore, one of the future works is to improve the HAR model's ability of distinguishing critical activities. Secondly, in our research, we only considered one option (1:1) of the possible ratio of real data and synthetic data. Given the ratio of real data and synthetic data could also have impacts on the performance of HAR model. Thirdly, the focus of the research is to improve the performance of HAR using ambient sensors. However, HAR in real world still face challenges of designing the sensor network layout and lifelong learning. The performance of ambient sensor based HAR not only relies on the algorithm recognizing activities but also is influenced by the design of the sensor network's layout. Hence, a suggested future research is to consider the influence of the design of the sensor network on the performance of HAR method. This would require researcher to consider the configuration of the sensor network and the mechanism of the HAR method simultaneously.

Lifelong learning refers to the model's ability of tackling concept drift (i.e., changes in the activity distribution, or the resident's living habit) and class evolution (i.e., the emergence of new activities in the future). Conventionally, a static model is not capable of lifelong learning.

Therefore, the second suggested future research is improving the HAR method the ability of learning new incoming data over time without forgetting the past data having learnt already.



## BIBLIOGRAPHY

- [1] Ranasinghe, S., Machot, F.A., & Mayr, H.C. (2016). A review on applications of activity recognition systems with regard to performance and evaluation. *International Journal of Distributed Sensor Networks*, 12.
- [2] Vallabh, P., & Malekian, R. (2018). Fall detection monitoring systems: a comprehensive review. *Journal of Ambient Intelligence and Humanized Computing*, 9, 1809-1833.
- [3] Bergmann, J.H., & McGregor, A. (2011). Body-worn sensor design: what do patients and clinicians want? *Annals of Biomedical Engineering*, 39, 2299-2312.
- [4] Chen, L., Hoey, J., Nugent, C., Cook, D., & Yu, Z. (2012). Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42, 790-808.
- [5] Guan, Y., & Plötz, T. (2017). Ensembles of Deep LSTM Learners for Activity Recognition using Wearables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1, 1 - 28.
- [6] Wang, J., Chen, Y., Gu, Y., Xiao, Y., & Pan, H. (2018). SensoryGANs: An effective generative adversarial framework for sensor-based human activity recognition. 2018 International Joint [7] Conference on Neural Networks (IJCNN), 1-8.
- [7] Li, X., Luo, J., & Younes, R. (2020). ActivityGAN: generative adversarial networks for data augmentation in sensor-based human activity recognition. *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*.
- [8] Ranasinghe, S., Machot, F.A., & Mayr, H.C. (2016). A review on applications of activity recognition systems with regard to performance and evaluation. *International Journal of Distributed Sensor Networks*, 12.
- [9] Banu, P. N., & Kavitha, R. (2020). Single activity recognition system: a review in internet of things (IoT), 257-271.
- [10] Mastorakis, G., & Makris, D. (2012). Fall detection system using Kinect's infrared sensor. *Journal of Real-Time Image Processing*, 9, 635-646.
- [11] Kim, K., Chalidabhongse, T. H., Harwood, D., & Davis, L. (2005). Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3), 172-185.
- [12] Stauffer, C., Grimson, W. (1999). Adaptive background mixture models for real-time tracking. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2, 246-52.

- [13] Elgammal, A., Harwood, D., & Davis, L. (2000). Non-parametric model for background subtraction. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1843, 751-767.
- [14] Kawatsu, C., Li, J., & Chung, C. J. (2013). Development of a fall detection system with Microsoft Kinect. In *Robot Intelligence Technology and Applications 2012*, 623-630.
- [15] Manzi, A., Moschetti, A., Limosani, R., Fiorini, L., & Cavallo, F. (2018). Enhancing activity recognition of self-localized robot through depth camera and wearable sensors. *IEEE Sensors Journal*, 18(22), 9324-9331.
- [16] Chen, W., & Chen, Y. (2017). An ensemble approach to activity recognition based on binary sensor readings. *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*, 1-5.
- [17] Yang, J., Nguyen, M., San, P.P., Li, X., & Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L., & Krishnaswamy, S. (2015, July). Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 3995-4001.
- [18] Cartwright, H. (2015). Artificial neural networks. *Methods in Molecular Biology*, 1260.
- [19] Neyshabur, B., Wu, Y., Salakhutdinov, R. R., & Srebro, N. (2016). Path-normalized optimization of recurrent neural networks with relu activations. In *Advances in Neural Information Processing Systems*, 3477-3485.
- [20] Zhang, Y., Wang, S., Ji, G., & Phillips, P. (2014). Fruit classification using computer vision and feedforward neural network. *Journal of Food Engineering*, 143, 167-177.
- [21] Jia, W., Muhammad, K., Wang, S., & Zhang, Y. (2017). Five-category classification of pathological brain images based on deep stacked sparse autoencoder. *Multimedia Tools and Applications*, 78, 4045-4064.
- [22] Wang, C. (2015). Time series neural network systems in stock index forecasting. *Computer Modelling & New Technologies*, 19(1), 57-61.
- [23] Dahikar, S. S., & Rode, S. V. (2014). Agricultural crop yield prediction using artificial neural network approach. *International journal of innovative research in electrical, electronics, instrumentation and control engineering*, 2(1), 683-686.
- [24] Ji, B., Sun, Y., Yang, S., & Wan, J. (2007). Artificial neural networks for rice yield prediction in mountainous regions. *The Journal of Agricultural Science*, 145(3), 249.
- [25] Ennett, C. M., Frize, M., & Charette, E. (2004). Improvement and automation of artificial neural networks to estimate medical outcomes. *Medical engineering & physics*, 26(4), 321-328.

- [26] Huang, W., Lai, K. K., Nakamori, Y., Wang, S., & Yu, L. (2007). Neural networks in finance and economics forecasting. *International Journal of Information Technology & Decision Making*, 6(01), 113-140.
- [27] Chung, K. C., Tan, S. S., & Holdsworth, D. K. (2008). Insolvency prediction model using multivariate discriminant analysis and artificial neural network for the finance industry in New Zealand. *International journal of business and management*, 39(1), 19-28.
- [28] Mayer, H., Gomez, F., Wierstra, D., Nagy, I., Knoll, A., & Schmidhuber, J. (2008). A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *Advanced Robotics*, 22(13-14), 1521-1537.
- [29] Xu, Y., Zhong, H., & Sheng, V. (2020). A novel privacy-preserving speech recognition framework using bidirectional LSTM. *Journal of Cloud Computing*, 9(1), 1-13.
- [30] Zhao, Z., Chen, W., Wu, X., Chen, P.C.Y., & Liu, J. (2017). LSTM network: A deep learning approach for short-term traffic forecast. *IET Image Processing*, 11(2), 68-75.
- [31] Graves, A., & Schmidhuber, J. (2008). Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in Neural Information Processing Systems*, 21, 545-552.
- [32] Ham, S. H., Ahn, H., & Kim, K. P. (2020). LSTM-based business process remaining time prediction model featured in activity-centric normalization techniques. *Journal of Internet Computing and Services*, 21(3), 83-92.
- [33] Kaji, D. A., Zech, J. R., Kim, J. S., Cho, S. K., Dangayach, N. S., Costa, A. B., & Oermann, E. K. (2019). An attention based deep learning model of clinical events in the intensive care unit. *PloS one*, 14(2), e0211057.
- [34] Zhao, Y., Wang, Y., Li, J., Liu, L., Ma, J., Zhong, Y., & Xia, Min. (2020). Airport arrival flow prediction considering meteorological factors based on deep-learning Methods. *Complexity*, 2020, Complexity, 2020, Vol.2020.
- [35] Saha, S. (2018, December). A comprehensive guide to convolutional neural networks — the ELI5 way. Retrieved November 2020, from <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- [36] Feng, Z., Mo, L., & Li, M. (2015). A random forest-based ensemble method for activity recognition. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, 2015*, 5074-5077.
- [37] Jurek, A., Nugent, C., Bi, Y., & Wu, S. (2014). Clustering-based ensemble learning for activity recognition in smart homes. *Sensors*, 14(7), 12285-12304.

- [38] Mo, L., Liu, S., Gao, R., & Freedson, P. (2012). Multi-sensor ensemble classifier for activity recognition. *Journal of Software Engineering and Applications*, 05, 113-116.
- [39] Xu, S., Tang, Q., Jin, L., & Pan, Z. (2019). A cascade ensemble learning model for human activity recognition with smartphones. *Sensors*, 19(10), 2307.
- [40] Mashhadi, P.S., Nowaczyk, S., & Pashami, S. (2019). Stacked ensemble of recurrent neural networks for predicting turbocharger remaining useful life. *Applied Sciences*, 10, 69.
- [41] Irvine, N., Nugent, C., Zhang, S., Wang, H., & Ng, W. (2020). Neural network ensembles for sensor-based human activity recognition within smart environments. *Sensors (Basel, Switzerland)*, 20.
- [42] Ashry, S., Ogawa, T., & Gomaa, W. (2020). CHARM-Deep: continuous human activity recognition model based on deep neural network using IMU sensors of smartwatch. *IEEE Sensors Journal*, 20, 8757-8770.
- [43] Ding, J., Li, X., & Gudivada, V. (2017). Augmentation and evaluation of training data for deep learning. *2017 IEEE International Conference on Big Data (Big Data)*, 2603-2611.
- [44] Xiao, Y., Wu, J., Lin, Z., & Zhao, X. (2018). A deep learning-based multi-model ensemble method for cancer prediction. *Computer Methods and Programs in Biomedicine*, 153, 1-9.
- [45] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7, 1-30.
- [46] Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*, 321, 321-331.
- [47] Guo, J., Li, Y., Hou, M., Han, S., & Ren, J. (2020). Recognition of daily activities of two residents in a smart home based on time clustering. *Sensors (Basel, Switzerland)*, 20.
- [48] Liciotti, D., Bernardini, M., Romeo, L., & Frontoni, E. (2020). A sequential deep learning application for recognising human activities in smart homes. *Neurocomputing*, 396, 501-513.
- [49] Liu, Y., Mu, Y., Chen, K., Li, Y., & Guo, J. (2020). Daily activity feature selection in smart homes based on pearson correlation coefficient. *Neural Processing Letters*, 1-17.
- [50] Lee, S., Yoon, S., & Cho, H. (2017). Human activity recognition from accelerometer data using Convolutional Neural Network. *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 131-134.
- [51] Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S. T., Tröster, G., Millán, J. D. R., & Roggen, D. (2013). The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15), 2033-2042.

- [52] Inoue, M., Inoue, S., & Nishida, T. (2018). Deep recurrent neural network for mobile human activity recognition with high throughput. *Artificial Life and Robotics*, 23(2), 173-185.
- [53] Yang, J., Nguyen, M., San, P.P., Li, X., & Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. *Yang, J. B.*,
- [54] Gochoo, M., Tan, T., Liu, S., Jean, F., Alnajjar, F.S., & Huang, S. (2019). Unobtrusive activity recognition of elderly people living alone using anonymous binary sensors and DCNN. *IEEE Journal of Biomedical and Health Informatics*, 23, 693-702.
- [55] Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S. T., Tröster, G., Millán, J. D. R., & Roggen, D. (2013). The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15), 2033-2042.
- [56] Liciotti, D., Bernardini, M., Romeo, L., & Frontoni, E. (2020). A sequential deep learning application for recognising human activities in smart homes. *Neurocomputing*, 396, 501-513.
- [57] Almeida, A., & Azkune, G. (2018). Predicting human behaviour with recurrent neural networks. *Applied Sciences*, 8, 305.
- [58] Wang, L., & Liu, R. (2020). Human activity recognition based on wearable sensor using hierarchical deep LSTM networks. *Circuits, Systems, and Signal Processing*, 39(2), 837-856.
- [59] Ronao, C., & Cho, S. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59(C), 235-244.
- [60] Yang, Z., Raymond, O., Zhang, C., Wan, Y., & Long, J. (2018). DFtNet: towards 2-bit dynamic fusion networks for accurate human activity recognition. *IEEE Access*, 6, 56750-56764.
- [61] Aiguo Wang, Guilin Chen, Jing Yang, Shenghui Zhao, & Chih-Yung Chang. (2016). A comparative study on human activity recognition using inertial sensors in a Smartphone. *IEEE Sensors Journal*, 16(11), 4566-4578.
- [62] Zhao, Y., Yang, R., Chevalier, G., Xu, X., & Zhang, Z. (2018). Deep residual bidir-ISTM for human activity recognition using wearable sensors. *Mathematical Problems in Engineering*, 2018, 13.
- [63] Singh, D., Merdivan, E., Hanke, S., Kropf, J., Geist, M., & Holzinger, A. (2017). Convolutional and recurrent neural networks for activity recognition in smart environment. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10344, 194-205
- [64] Yang, H., Gong, S., Liu, Y., Lin, Z., & Qu, Y. (2020). A multi-task learning model for daily activity forecast in smart home. *Sensors (Basel, Switzerland)*, 20.

- [65] Gochoo, M., Tan, T., Liu, S., Jean, F., Alnajjar, F.S., & Huang, S. (2019). Unobtrusive activity recognition of elderly people living alone using anonymous binary sensors and DCNN. *IEEE Journal of Biomedical and Health Informatics*, 23, 693-702.
- [66] Zheng, X., Wang, M., & Ordieres-Meré, J. (2018). Comparison of data preprocessing approaches for applying deep learning to human activity recognition in the context of industry 4.0. *Sensors*, 18(7), 2146.
- [67] Li, F., Shirahama, K., Nisar, M. A., Köping, L., & Grzegorzec, M. (2018). Comparison of feature learning methods for human activity recognition using wearable sensors. *Sensors*, 18(2), 679.
- [68] McCulloch, W., & Pitts, W.F. (1990). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52, 99-115.
- [69] Greff, K., Srivastava, R., Koutník, J., Steunebrink, B., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28, 2222-2232.