MSc AI Group Project - Human Activity Recognition

— Final Report —

Lew Hwi Hsien Astrid, Joanna Ye, Alexandra Embs, Chia-Hsin Lin, Yuhan Wang, Priya Jain
{hl1022, jdy15, ane21, cl2222, yw6722, pj22}@imperial.ac.uk

Supervisor: Prof Julie McCann, Huang Shuokang
Module: COMP70048, Imperial College London

November 19th, 2023

1 Introduction

In this section, we define Human Activity Recognition (HAR), the impetus for our research, and our group's contribution on HAR research. We also review relevant past works.

1.1 Human Activity Recognition

Human Activity Recognition (HAR) aims to recognise human activities such as walking, running and jumping in indoor environments. HAR has gained its importance due to various applications in both academic and industrial settings. For example, elderly people monitoring [3], energy efficient systems for smart buildings [6], and smoking detection [10].

1.2 Impetus

To detect human movements, WiFi-based methods have gained popularity due to the ubiquitousness of WiFi machines and the "device-free" perspective of the user. WiFi-based methods are based on the principle that different human movements result in unique signal patterns between the transmitter and the receiver [8]. There are two kinds of WiFi-signal: Received Signal Strength (RSS) and Channel State Information (CSI). While CSI can only be obtained via certain WiFi devices, it is preferred in the majority of research because it contains richer information than RSS.

However, to the best of our knowledge, most of the CSI-based HAR research only focuses on recognising activities for a single person, which imposes limitations on real-life applications. For example, in reality more than one person is likely to be in the same room. The presence of additional people hinders the prediction performance of the models, as shown by previous research [10]. Moreover, we may be interested in the movements of multiple people at the same time. For example, for fall detection, we need to detect movements of all people in the room to make sure no one is falling. One-person activity recognition is difficult to be applied in this case and most of the real-life applications.

To address this issue, in this project we conduct novel research on detecting human movements for more than one person in the same environment. Specifically, we focus on recognising activities of two people at the same time, which includes collecting two-person data and constructing classification models to detect the movement of each person simultaneously. After model construction, we evaluate the model performance through cross-validation and present insights found for multi-target HAR.

1.3 Group Contribution

Based on previous research, we consider three types of model architectures for Human Activity Recognition: Long-Short Term Memory (LSTM) [2], Convolutional Neural Networks (CNN) [5], and Two-Stream Convolution Augmented Transformer (THAT)[4]. We achieve state-of-the-art results for both one-person and two-person activity recognition, where one-person data is collected to observe benchmark performance of the previous models on our own dataset. The result shows that the THAT model excels at all types of HAR, including both one-person and two-person activity detection and the case where the number of people performing activities is unknown in advance.

The following summarises our contributions:

- Construct a first-of-its-kind dataset for multi-target (two-person) activity recognition in clean and noisy room environments.
- Achieve state-of-the-art results on multi-target (two-person) activity recognition through the development of neural network models.
- Analyse and compare Human Activity Recognition results derived from three different neural network architectures (ranging from 88.6% for LSTM to 94.8% accuracy for THAT).

1.4 Past Work

To conduct two-person HAR research, we review relevant past works on one-person HAR. The past works can be divided into two areas: 1) Publicly-available datasets and 2) Deep learning models used to recognise human activities via raw CSI signal.

1.4.1 Public Dataset

CSI-based HAR requires the combination of specific hardware and software, so there are only a few public datasets available. The most cited dataset consists of 7 types of activity (lie down, fall, walk, run, stand up, sit down, pick up) taken from 7 people separately in an office room [9].

1.4.2 Deep Learning Models

Instead of considering the traditional machine learning (ML) models which require hand-crafted features, we review deep learning models which process raw CSI data. Two types of information are presented in raw CSI signals: channel and temporal information. Channel information contains the amplitude and frequency of the signal, and temporal information records the specific timing of each signal point.

One way to represent raw CSI signals is to convert them into two-dimensional greyscale images; the resulting images then act as inputs for the CNN [5]. The model disregards temporal information from the signal and processes all channel information in parallel. The human activities are identified through image features such as colour and texture.

To include sequential information from the input signal, the LSTM network is proposed to directly process raw CSI signals [2]. Since different portions of the input sequence may contribute differently to the final activity recognition, attention-based bidirectional LSTM (ABLSTM) is used to assign different weights to the learned features along input sequences [2].

To learn both "time-over-channel" and "channel-over-time" features simultaneously, the THAT model is proposed [4]. The augmented "time-over-channel" data, which LSTM models do not include, enhances the model's ability to learn temporal information better according to the authors [4]. The self-attention mechanism in both channel and temporal streams overcomes the long range dependency problem of LSTMs and allows the model to focus on multiple parts of input sequences.

This introduction section summarised the HAR definition, problem statement, relevant past works and our group's contribution in this area of research. Specifically we have constructed the first multi-target dataset for HAR and assessed the performance of three different neural network architectures for the prediction task.

2 Data Collection

This section documents the data collection plan and implementation process. It covers the environments considered, equipment used and activity types investigated.

2.1 Data Collection Details

In this project, we collected one- and two-person WiFi CSI data for five activities: sit down, stand up, walk, run, and jump. Two environments are considered: a clean room and a noisy room. The clean room is a small office room with only the equipment necessary for data collection. The noisy room is a ten-person office room with standard office equipment, including desktops transmitting signals irrelevant to HAR, to replicate the real-life setting. We used Intel 5300 wireless NIC with 2.4 GHz WiFi frequency and two Linux desktops for sending and receiving signals. To make our JPC3A dataset more complete, reliable, and diverse, we repeat each activity in each environment 120 times with a sampling rate of 1000 Hz. For the two-person data, we consider all two-activity combinations chosen from the five activities, leading to 10 combinations in total in the resulting two-person dataset. The final dataset consists of the raw CSI signals and their corresponding activity labels.

The collected data was saved in files with a defined naming convention, such as `csi_n_1p_run_001.dat` and `csi_c_2p_sit_run_001.dat`. The former contains the first sample of one person running in the noisy room. The latter contains the first sample of one person sitting down and one person running in the clean room. The dataset is called **JPC3A**. Figure 1 summarises the data collection process and figure 2 shows the two environments considered in the data collection.

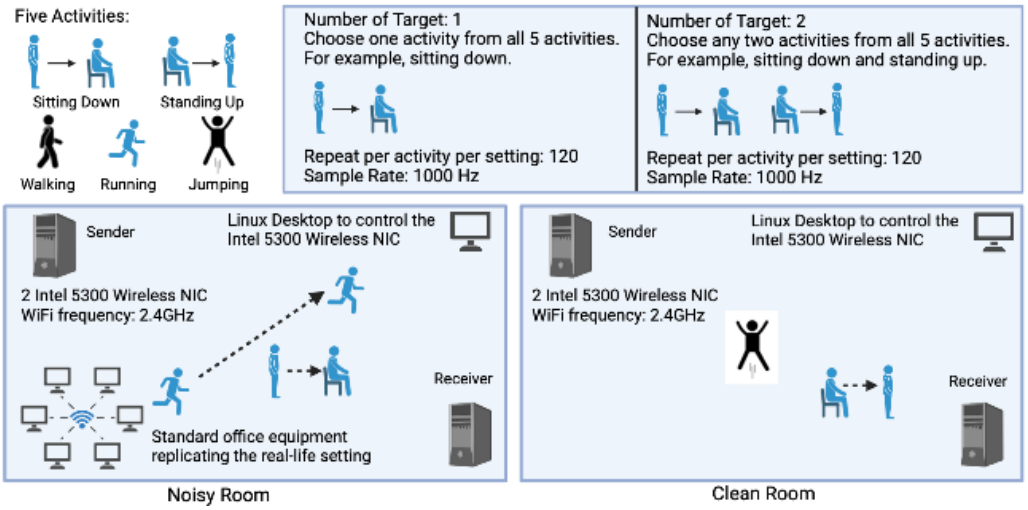

Figure 1: Summary of the data collection process.

Figure 2: Pictures of the clean room (left) and noisy room (right).

This section covered the data collection plan and process. It also illustrated the two environments considered, equipment used and activity types investigated.

3 Technical Decisions, Design and Architecture

As stated previously in Section 1.4.2, three different network architectures were developed in an effort to tackle this activity classification problem. While the individual architectures of the LSTM, CNN, and THAT differed, a consistent raw data processing, model building objective, and model evaluation procedure was adopted amongst all three networks. This section covers the technical decisions behind the common model development methodologies across the three different neural network architectures.

3.0.1 Raw Data Processing

Prior to individual model preprocessing, an overall raw data processing was carried out for all three architectures. The raw CSI amplitude for each distinct activity recording was imported as a CSI matrix of dimension N x T x P. In this case, N is the number of samples, T represents the time dimension with each row entry signifying each time step sampled at 1000 Hz, and P represents the CSI feature vector

size. It is important to note that for the public dataset, P = 90 as the experimental set up included 3 transmission antennas and 30 subcarriers. On the other hand, for the JPC3A dataset, P = 270 as the experimental set up included 3 transmission antennas, 3 receiving antennas, and 30 subcarriers.

The data was then transformed into 1D vectors. In the case of the public dataset, the data was reshaped into size 1x90000 by applying a sliding window which concatenated 1000 time steps of the 90 dimensional rows. In the case of JPC3A dataset, each sample was also concatenated into size 1x(270*T) where T, once again, is the number of time steps that occurred for that particular data sample. It is important to note that for the JPC3A dataset, T varies per data sample.

Lastly, the corresponding activity labels were constructed for each of these 1D vectors. In the case of the public dataset, this included a 1x7 dimensional vector, as this dataset included seven activities. For the JPC3A dataset, the activity labels had dimension 1x5, as five activities were considered.

3.0.2 Model Building

All three network architectures were constructed using the PyTorch library. As an initial starting point, the CNN, LSTM/ABLSTM, and THAT architectures described in Moshiri et al., Chen et al., Yousefi et al., and Li et al. were implemented and verified on the one person public dataset provided by Yousefi et al. [2], [4], [5], [9]. Once the performance levels matched or exceeded those of previous findings, the networks were adapted and tested on the JPC3A dataset.

In total five different models per CNN, LSTM, and THAT architecture were developed using the JPC3A dataset: 1p Clean, 1p Noisy, 2p Clean, 2p Noisy, and Combined. Therefore, in total, fifteen models were constructed. 1p Clean and 1p Noisy are the models constructed using the one person recorded activities in the clean and noisy rooms respectively. These models only predict a single activity per input. Likewise, 2p Clean and 2p Noisy are the models constructed only using the two person recorded activities in the clean and noisy rooms respectively. These models predict two activities per input. Lastly, a Combined model which utilized both one-person and two-person data as input to the model was developed. The goal of this model was to not only be able to decipher which activity was occurring, but to determine how many individuals were in the recording. The development of this final combined model for each of the network architectures was the ultimate aim of this project.

3.0.3 Model Evaluation

Each of the fifteen models were developed and tuned individually. For the final evaluation of the combined models, a five-fold cross-validation was implemented. Accuracy was used as the main evaluation metric, and Precision and Recall metrics were also analyzed.

3.1 LSTM and Attention-based bidirectional LSTM (ABLSTM)

3.1.1 Model Architecture

Figure 3 shows the overall architecture for the final simple LSTM and attention-based bidirectional LSTM models. The LSTM class in PyTorch is used to implement the standard LSTM equations [1]. The simple LSTM model processes the CSI signal sequentially in forward order only. The final hidden state is then passed through a linear layer with output dimensions equal to the number of possible activities (five for the JPC3A dataset).

The ABLSTM model extends the simple LSTM model in a number of ways. Firstly, the LSTM is bidirectional, processing the input in both forward and reverse order. Secondly, the hidden states from all time steps are used for subsequent prediction, rather than just the final hidden state. These hidden states are passed through an attention mechanism (described in more detail in Section 3.1.3) before then being passed through a linear layer.

Finally, an output activation function is applied to the logits to predict which activity/activities are present. For the one-person dataset where each sample has just one label, a softmax activation function is used and the model is trained with cross-entropy loss. For the two-person and combined datasets where each sample can have one or more labels, a sigmoid activation function is applied and

the model is then trained with binary cross-entropy loss. This treats the multi-target problem as a multi-label classification problem, so that it can be applied to data with an arbitrary number of activity combinations without needing to change the output dimensions.

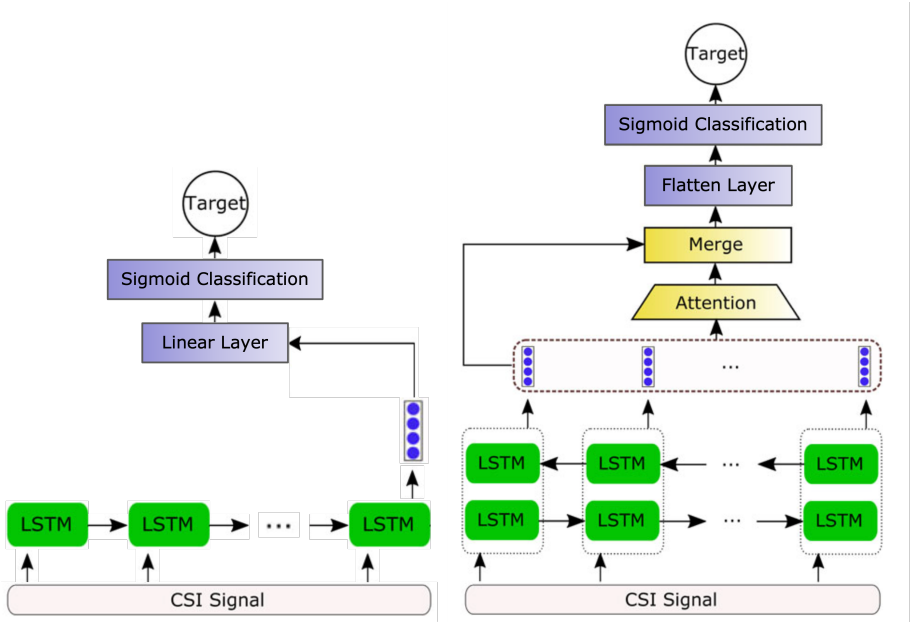

Figure 3: LSTM (left) and ABLSTM (right) model architectures (adapted from Chen et al. [2]).

3.1.2 Pre-processing

For the public dataset, no further pre-processing was required beyond that described in Section 3.0.1. In order to input the data into the LSTM, each data point was reshaped into a 1000x90 tensor, where 1000 is equal to the window size/number of time steps, and 90 is the feature vector size.

For the JPC3A dataset, each data point has a different number of time steps depending on the number of WiFi packets received out of the 4000 transmitted (this number is usually between 3000 and 4000). It was decided to reduce the number of time steps to a fixed number to be able to input batched data into the LSTM without the need for padding, and to reduce the computational complexity of the model. A value of 1000 was chosen based on the window size for the public dataset.

Two methods were investigated to reduce the number of time steps to 1000. The first used the `signal.resample` function from `scipy` to resample each data point from a variable number of time steps to a fixed length of 1000. The second took the first 3000 time steps from each data point, and then applied average pooling with a kernel size of 3 to reduce this to 1000 time steps. It was found that the first method gave slightly better results for the simple LSTM, hence this pre-processing method was used for all of the results reported below.

3.1.3 Attention Mechanisms and Pooling

Two different attention mechanisms were investigated: scaled dot-product attention [7] and simple attention [2]. See Figure 4 for details on their implementation.

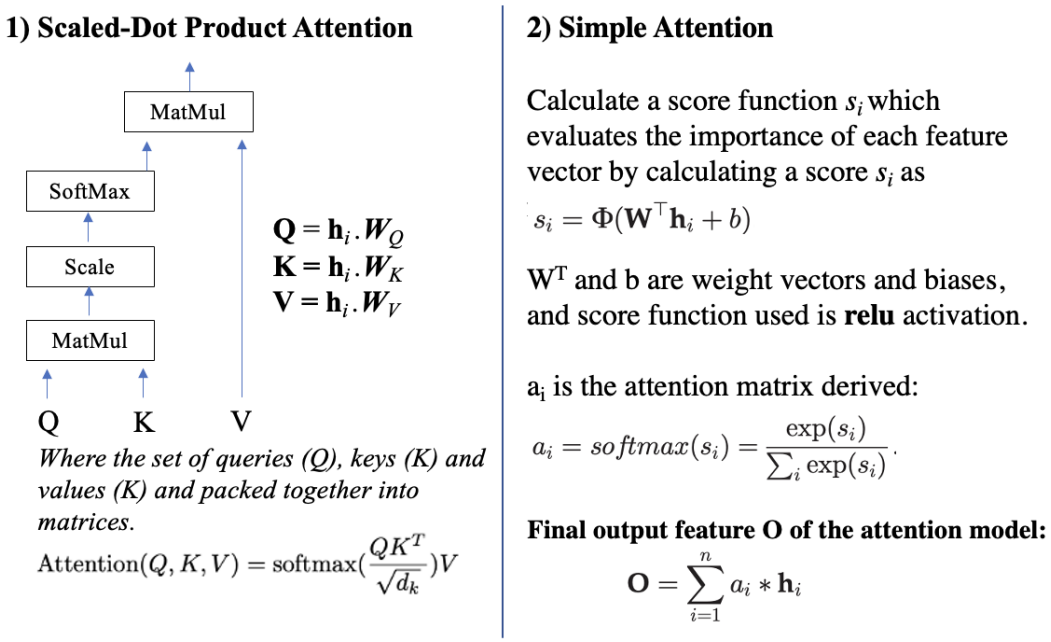

Figure 4: Two attention mechanisms investigated, adapted from [7] and [2].

Both attention mechanisms yielded comparable results for the different datasets, with simple attention slightly outperforming scaled dot product attention for most datasets and parameter combinations by an average of approximately 1% accuracy. This is surprising since an additional pooling factor was imposed for simple attention; one possible reason for this could be due to a higher number of parameters introduced into the simple attention model which helped to improve its representation.

For simple attention, the attention matrix dimensions were too large for the GPU's memory capacity. Therefore, we also experimented with different pooling techniques, namely max vs. average pooling, and pooling before LSTM input vs. before attention. Based on the preliminary results, average pooling just before the attention mechanism provided the best accuracy results. Therefore, subsequent hyperparameter tuning was done using average pooling just before the attention mechanism. A pooling factor of 10 and 20 were required for the 1-person and 2-person datasets respectively. It should be noted that for the public dataset, both average pooling before feeding into the LSTM and before the attention mechanism had to be employed to enable the model to run with the GPU capacity available.

3.1.4 Hyperparameter Tuning

For both models, we used one hidden layer as this was able to capture sufficient characteristics of the data and provide reasonable results. We tuned several hyperparameters including the number of epochs, learning rate and batch size. We also tuned the hidden dimension within a range that provided reasonable results without making the models too large to train.

The best tuning results are documented in Table 1 and Table 2 below. We found that a larger hidden dimension was required for the JPC3A dataset compared to the public dataset, and suspect that this may be due to the larger feature vector size for the JPC3A dataset (270 compared to 90). Note that the accuracy of the simple LSTM on the public dataset exceeds that reported in Chen et al. [2] by approximately 7%. As expected, the ABLSTM consistently outperforms the LSTM, and higher accuracies are achieved on the clean datasets.

3.2 CNN

3.2.1 Model Architecture

The initial architecture is based on the architecture proposed by Moshiri et al. [5]. This includes two convolution layers, with max pooling layers and batch normalisation. Batch normalisation ensures

Dataset	Learning Rate	Batch Size	Hidden Dimension	Epoch No.	Accuracy
Public	1e-4	5	200	200	97.0%
1p clean	2e-5	10	300	600	92.5%
1p noisy	1e-5	10	400	1000	89.2%
2p clean	1e-5	10	300	2000	90.4%
2p noisy	2e-5	10	400	600	84.5%
Combined	1e-5	10	400	1000	88.6%

Table 1: Hyperparameter tuning results for the simple LSTM.

Dataset	Att/Pool	LR	Batch Size	Hidden Dim	Epoch No.	Accuracy
Public	Simple/k=5,4	1e-5	10	200	300	99.3%
1p clean	Simple/k=10	1e-5	10	400	50	95.0%
1p noisy	Simple/k=10	5e-6	10	400	50	89.2%
2p clean	Scaled Dot/k=1	1e-5	10	400	100	91.6%
2p noisy	Simple/k=20	1e-5	50	400	100	88.1%
Combined	Simple/k=20	1e-5	30	300	100	90.8%

Table 2: Hyperparameter tuning results for the ABLSTM.

the training process is stabilised and contributes to capacity control. Max pooling is used to provide local, approximate shift invariance and extract important features. The data is then passed through two fully connected layers for classification, and one dropout layer is included between these layers to prevent over fitting and improve generalisation of the model. Finally, the final output is passed through a SoftMax activation function and cross entropy is used for the loss function. Initial experiments were done on the full public data set using this architecture, and this model achieved an accuracy of 90%.

This architecture was trialled on the JPC3A dataset and produced poor results. To rectify this, a more complex CNN architecture was designed. This involved increasing the number of convolution layers to increase the model capacity and to allow the model to learn a better hierarchical representation. Furthermore, the activation function was changed from ReLU to Leaky ReLU, as this will prevent the dying neuron problem. Max pooling layers and batch normalisation are retained after each convolution layer for the same reasons outlined in the initial architecture. Four layers was selected for one person models, however five layers is utilised for two person models, as further model capacity was required to handle multi-target data. Finally, the same five layer model is utilised for the combined model. This improved architecture is visualised in Figure 5 and the details of the convolution and max pooling layers are outlined in Table 3. Following the convolution layers, the data is fed into two fully connected layers, the first layer contains 8192 in features and 512 out features, and the second layer contains 512 in features and 5 output features, each corresponding to an activity label. This dense network also contains one dropout layer for regularisation. A sigmoid activation function is used for the final output as this model needs to be able to deal with a multi-label classification whilst handling two person data. Therefore, binary cross entropy is selected as the loss function.

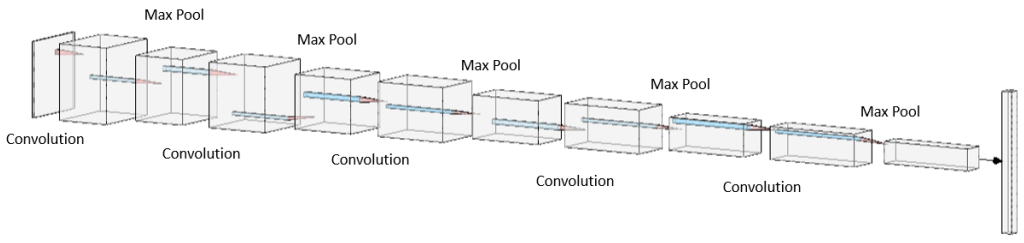

Figure 5: CNN architecture visual - final model for the JPC3A dataset.

Layer	Kernel Size	In Channels	Out Channels	Padding	Image Size
Convolution Layer 1	3	1	32	1	128
Max Pooling Layer 1	2	32	32	0	64
Convolution Layer 2	3	32	64	1	64
Max Pooling Layer 2	2	64	64	0	32
Convolution Layer 3	3	64	128	1	32
Max Pooling Layer 3	2	128	128	0	16
Convolution Layer 4	3	128	256	1	16
Max Pooling Layer 4	2	256	256	0	8
Convolution Layer 5	3	256	512	1	8
Max Pooling Layer 5	2	512	512	0	4

Table 3: Convolution and max pooling layer details of final model for the JPC3A dataset.

3.2.2 Pre-processing

In addition to the raw data preprocessing described in section 3.0.1, each data sample was converted back into its CSI matrix form. In the case of the public dataset this was 1000x90 and in the case of the JPC3A dataset this was Tx270 where T is the time samples. To address the variable time length issue in the JPC3A dataset, the maximum time length M of all the samples was determined, and each CSI matrix was padded with zeros to obtain dimensions of Mx270.

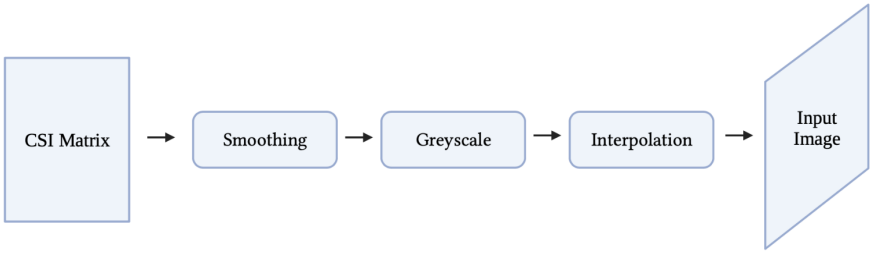

Figure 6: CNN preprocessing pipeline.

The CSI matrix was then smoothed using average pooling and a kernel size of 11. Following this, the matrix was converted to greyscale by uniformly normalising the pixel values between 0 to 255. Finally the image was interpolated to dimensions of 128x128. Computational cost and information preservation were considered when selecting an interpolation technique. Bilinear interpolation was selected as this considers four nearest neighbours and averages these, which preserves more information than simple nearest neighbour interpolation which simply copies the single nearest pixel value. This method is also computationally efficient as compared with more complex approaches such as bicubic interpolation. This singular square image is the input data format to the CNN model. This preprocessing technique remained consistent across all five CNN models developed.

3.2.3 Hyperparameter Tuning

To begin the hyperparameter tuning process, an initial investigation into the influence of image size on model performance was conducted. Based on the results in Table 4, an image size of 128 was found to have the best performance for models trained on the one-person and two-person datasets. As a result, an input image size of 128 was fixed for all the models. Furthermore, the one person model with a large image size of 256 performed very poorly with only 14.5 % accuracy. We suggest that this low accuracy is likely a result of overfitting; it can be argued that an input image of size of 256 is too large resulting in too many model parameters. In other words, the model cannot generalise well and the model is capturing information overly specific to the training data. Note that the two person model with image size of 256 did not suffer from significant degradation in performance; this is potentially due to the two person data containing more information as well as noise, thus the two person model is not suffering

from overfitting with this image size. Future work could include conducting experiments to verify this theory to gain a better understanding of the relationship between image size and model overfitting for HAR. Nevertheless, 128 image size results in highest model accuracy for one- and two-person data. Therefore, based on our results it is clear 128 is the best image size for all models developed.

Dataset	Image Size	Accuracy
1p	64/128/256	85.4%/86.5%/14.5%
2p	64/128/256	87.4%/87.9%/86.0%

Table 4: Image size - hyperparameter tuning results for CNN.

In addition to this, several other hyperparameters were tuned including learning rate, batch size, weight decay, and number of epochs. The finalized parameters of each of the five models for the JPC3A dataset, as well as the model trained on the public dataset, can be found in Table 5.

Dataset	Learning Rate	Batch Size	Weight Decay	Epoch No.	Accuracy
Public	0.001	32	1e-5	60	90.0%
1p Clean	0.001	32	1e-4	50	89.9%
1p Noisy	0.001	64	1e-5	30	90.8%
2p Clean	0.001	32	1e-4	20	88.0%
2p Noisy	0.001	32	1e-4	10	87.7%
Combined	0.001	16	1e-5	30	89.1 %

Table 5: Hyperparameter tuning results for the CNN.

3.2.4 Data Augmentation

Data augmentation is an effective technique for artificially increasing the size of the training data set. This was trialled on the JPC3A dataset to understand whether this would further increase model accuracy. The augmentation techniques utilised include small random rotations, small random shifts and randomly applying Gaussian blur. Random rotations and shift should mimic the activity occurring at different time points in the time window of the data sample collected, and Gaussian blur could mimic a noisier environment. Optimal hyper-parameters were utilised and the dataset was doubled in size with augmentation. The model was trained on more epochs than previously selected due to the larger training dataset size. For 50, 75 and 100 epochs the respective accuracies are 85.9%, 87.0% and 88.1%. Therefore, it is clear that data augmentation does not provide any benefit to the model performance. We suggest that this is due to the dataset already containing sufficient noise and variability. Specifically, data was collected in different environments (clean and noisy) and different individuals are performing activities in varying combinations, inevitably leading to variability in the training data. This could be the reason augmenting the data provides no additional information to the model.

3.3 Transformer

3.3.1 Model Architecture

Figure 7 (left) shows the architecture of the THAT model [4]. The model is designed to overcome shortcomings of the CNN and LSTM models. This section only provides intuition of how each model component helps classify human activities from raw input signals. For full implementation details, please refer to the original paper [4].

The model architecture consists of four layers: preprocessing, Multi-scale Convolution Augmented Transformer (MCAT), aggregation, and prediction layers. The preprocessing layer converts raw signals with arbitrary length into fixed length sequences via time alignment method. After timing alignment, the layer then separates each input sequence into two streams: temporal stream and channel stream, where each stream extracts “channel-over-time” and “time-over-channel” features respectively.

Channel-over-time features capture channel patterns along time, similar to the features generated by the LSTM in Section 3.1. However, unlike the LSTM, which uses every time step of the input signal to encode temporal information, the THAT encodes temporal information through only a few "time ranges" to reduce noise, where ranges are defined by a number of Gaussian distributions. Details of Gaussian range-based encoding are shown on the right of Figure 7. On the other hand, time-over-channel features, which are not considered in both the CNN and LSTM, capture time patterns over different channels to help distinguish activities with similar movement but different body parts.

Multi-scale Convolution Augmented Transformer (MCAT) layers are used to extract desired features in both streams. Each stream contains a stack of MCAT layers. Each MCAT layer consists of two sub-layers: multi-head self-attention and multi-scale CNN with adaptive scale. Multi-head self-attention generates hidden representations of whole input sequences, aiming to overcome the vanishing/exploding gradient problem in the LSTM. On top of multi-head attention, the multi-scale CNN then extracts local features from the (global) hidden representation given by multi-head attention with multiple kernel sizes. Unlike the CNN in Section 3.2 which weights all features given by different kernel sizes equally, in THAT, features generated with different kernel sizes are weighted by an adaptive attention mechanism, which assigns larger weights to features produced by suitable kernel sizes. To ensure stable gradient flow, residual connections and layer normalisation are added after each sub-layer.

To make activity prediction(s), features learnt from the two streams are first max-pooled to extract the most important information and then concatenated in the aggregation layer. The concatenated features are then passed into the prediction layer, which consists of a fully-connected network generating logit scores for each activity and a final classifier to output prediction.

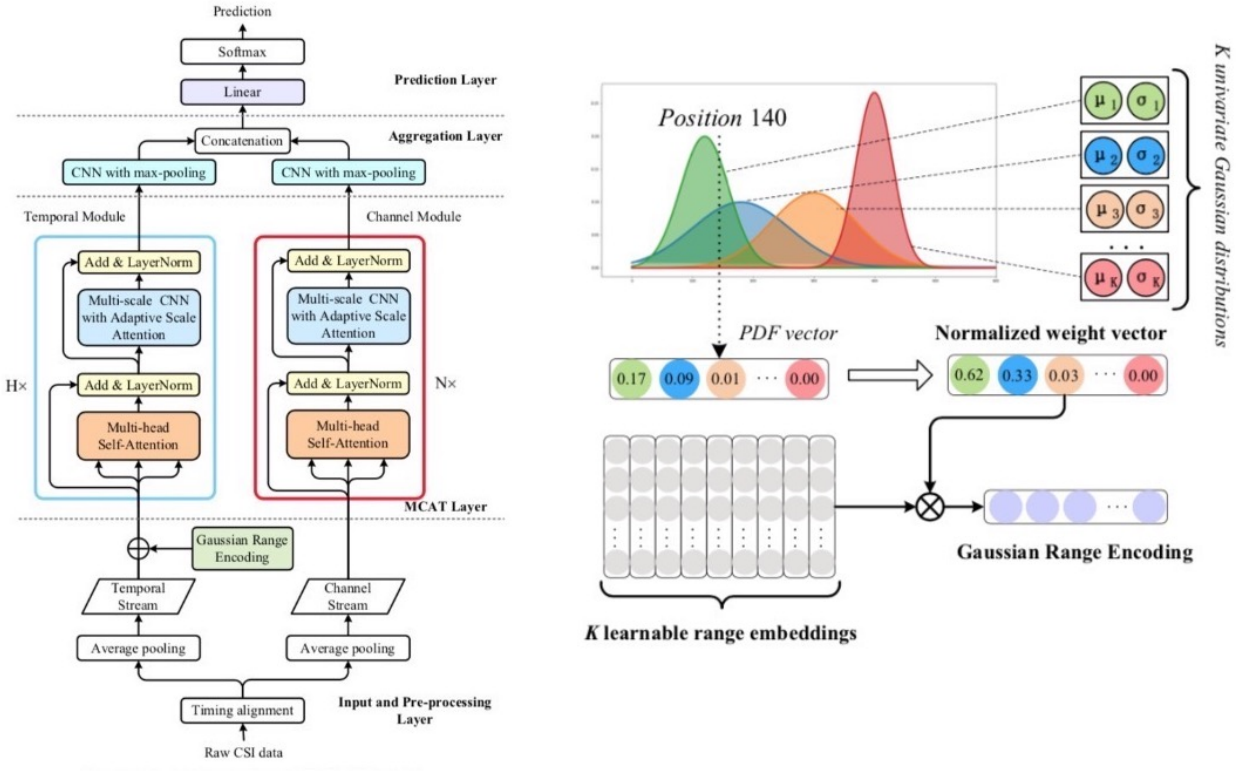

Figure 7: Transformer architecture (left) and range-based positional encoding (right) [4].

3.3.2 Pre-processing

In the preprocessing layer, we transform the CSI data into fixed length to be used as input for the MCAT layer. Each raw CSI sample is recorded in a t by C matrix, where t is the number of time steps varying by each sample and C is a fixed number of channels from the WiFi machines.

To enable batch processing, all input signals need to have the same temporal length, i.e. same t. We reshape raw inputs with timing alignment technique, where the temporal length t of each sample is evenly divided into a pre-defined number of slots T, and the channel information of each sample is mapped to its corresponding slot based on its time stamp. This ensures each record has the same dimensionality $T \times C$. To reduce memory usage, the temporal dimension T is further compressed by averaging the adjacent time slots (average pooling), resulting in a smaller temporal dimension T.

To extract channel-over-time features, data in dimensionality $T \times C$ is used as input for the temporal stream. For the channel stream, data in $C \times T$, which can be simply obtained from transposing the $T \times C$ data, is used to extract time-over-channel features.

3.3.3 Changes Adopted for Multi-activity Recognition

To perform multi-target HAR, we applied several changes to the preprocessing, MCAT, and prediction layers of the THAT model.

Compared to the public dataset which the original THAT model is developed on, the JPC3A dataset samples have much shorter temporal length. Therefore we shorten and experiment with different proposed input lengths T in the preprocessing layer for different datasets.

To make a multi-activity prediction, we replace the final softmax classifier in the prediction layer by multiple sigmoid functions, which consider a certain activity as present if the corresponding sigmoid output is greater than 0.5. We also change loss function from negative log likelihood loss (`torch.nn.NLLLoss`) to binary cross entropy loss (`torch.nn.BCELoss`) for each sigmoid classifier.

Since we have a shorter input length for each sample and the nature of two-person data is different from that of one-person data, in the MCAT layers we experiment with a different number of attention heads for multi-head self-attention and both the number and size of kernels in multi-scale CNN. In addition to components inside the MCAT layers, we also experiment with different numbers of layers of MCAT in both the temporal and channel streams of the model.

Inspired by the LSTM model, we also tried the `resample` function from `scipy` to reshape inputs into a fixed length in the preprocessing layer. However, unlike the LSTM, we found the training performance became unstable. Therefore, we kept the time alignment method developed in the original paper.

Except for changes due to the nature of the two-person dataset, we also tuned other hyperparmaters such as learning rate and batch size for better training process. More hyperparameters and the results are discussed in Section 3.3.4.

3.3.4 Hyperparameter Tuning

The hyperparameters we considered are learning rate, batch size, further pooling factor for temporal length T, number of epochs, temporal (horizontal) transformer layers, temporal (horizontal) transformer heads, channel (vertical) transformer layers, channel (vertical) transformer heads, pre-defined number of slots in time alignment technique in preprocessing (input length). We also tuned the number and size of kernels in the multi-scale CNN in the MCAT layer for both temporal and channel streams and clipped gradient for better training.

We found that including more types (sizes) of kernel in the multi-scale CNN increases the model performance, so we increased the number of kernel sizes from two to three, or even four, to learn more distinct features. For two-person noisy data, the kernel sizes on the temporal stream are set to 10, 35, 70, and 115, and those for the channel stream are set to 2, 8, and 12. For combined data, we set kernel sizes to 10, 35, 70, and 100 on the temporal stream and kept those in the channel stream the same as 2, 8, 12. These combinations give better performance according to our experiments.

In addition to kernel sizes, we also increased the number of kernels in the CNN in the temporal stream from 128 to 256, which enables the model to learn more features. Due to the GPU capacity, we can only run the model with up to four different sizes of kernel on temporal and channel streams. Sometimes we need to use a smaller input length (larger pooling factor) such as 7 or 8 on the temporal side if the proposed input length is too long and multiple kernel sizes are considered.

However, similar to the CNN in Section 3.2.3, longer input length does not necessary lead to higher accuracy. For example, for one-person noisy data, setting input length to 350 performs much better than setting it to 2000 (93.7% vs 75.8% in accuracy). We suspect longer input lengths lead to over-fitting.

The final results and hyperparameters set for the different datasets are documented in table 6.

Dataset	Lr	Batch	Sample	Epoch	Hlayers	Hheads	Vlayers	Vheads	Input length	Accuracy
Public	1e-3	16	4	50	5	9	1	200	2000	98.7 %
1p clean	1e-3	16	8	50	9	27	6	100	1000	94.2 %
1p noisy	5e-4	8	1	50	7	10	1	50	350	93.7 %
2p clean	1e-3	16	4	200	9	27	6	100	1000	94.8 %
2p noisy	1e-3	16	7	200	9	27	6	350	700	94.4 %
Combined	1e-4	5	7	70	6	27	1	100	700	94.8 %

Table 6: Hyperparameter tuning results for THAT.

3.4 Software Engineering Practices

In this section, the software engineering practices that were adhered to throughout the project implementation will be highlighted. In addition, instructions pertaining to running any of the developed work will be detailed in section 3.4.2.

3.4.1 Scrum Approach

While this project is more research focused than conventional software engineering projects, standard software engineering practices were implemented. In this particular case, scrum practices were adopted with one week sprints.

Each Friday, sprint planning meetings were held to devise which objectives were to be completed in the coming week. The following Friday, a combined sprint review and sprint retrospective meeting was held in which each group reported their results for the week as well as any set backs that occurred. This information was then used to develop the sprint plan for the following week. Biweekly meetings with the group supervisor and PhD mentor also acted as sprint review/retrospective meetings as the team further discussed how to improve practices and planning. In addition to this, it should also be noted that each of the three models was worked on in pairs in an effort to ensure all individual work could be verified by a second member.

While scrum is an unconventional approach to research based projects, the group found the weekly sprint set up was beneficial in developing the three models in parallel.

3.4.2 Version Control Repositories

For this project, GitLab was utilized as the main version control system. A different repository has been constructed for each of the three model architectures as well as the raw data preprocessing pipeline. The link to each of these respective models can be found in the URL text file submitted separately. Instructions regarding running the individual models can be found in each of the respective README.md files.

4 Evaluation

This section highlights the final model performances and insights. It also covers the overall evaluation on our team's contribution and efforts.

4.1 Final Model Performances

Table 7 summarises the accuracies of the final models on the combined dataset, for each activity individually as well as the overall average. Tables 8 and 9 summarise the recall and precision values

as an indication of the number of false negatives and false positives per activity.

Model	Jump	Run	Sit	Stand	Walk	Overall
LSTM/ABLSTM	90.9/91.7	86.8/89.5	90.0/90.1	88.5/91.2	86.9/91.4	**88.6/90.8**
CNN	91.0	88.3	87.4	88.2	90.3	**89.0**
THAT	95.2	94.8	94.1	94.5	95.3	**94.8**

Table 7: Accuracy values of the final models (%) for each individual activity and overall.

Model	Jump	Run	Sit	Stand	Walk	Overall
LSTM/ABLSTM	85.8/82.8	77.1/79.1	82.3/83.4	76.1/83.2	80.1/81.6	**80.3/82.0**
CNN	89.3	85.1	79.1	75.3	83.6	**82.5**
THAT	94.9	96.4	94.8	94.9	96.0	**95.4**

Table 8: Recall values of the final models (%) for each individual activity and overall.

Model	Jump	Run	Sit	Stand	Walk	Overall
LSTM/ABLSTM	86.8/91.6	82.1/88.2	86.9/86.5	87.6/89.6	80.6/91.7	**84.8/89.5**
CNN	84.7	81.4	82.4	87.7	87.4	**84.7**
THAT	98.0	95.8	96.3	96.9	97.0	**96.8**

Table 9: Precision values of the final models (%) for each individual activity and overall.

4.1.1 LSTM and ABLSTM

- The precision of the LSTM and ABLSTM models are higher than its recall, suggesting that the LSTM and ABLSTM are more susceptible to false negatives (failing to recognise an activity when it is present) than false positives (wrongly predicting an activity as present).
- The LSTM is more accurate at recognising activities that are done in one location (jump, sit and stand), compared to activities that involve changing location (run and walk). Closer analysis of the false positives suggest that this may be because of the high number of false positives for run when walk is present, and vice versa. It is hypothesised that these two activities are easily confused as they involve very similar movements and mainly differ only in the speed of the movement.
- For ABLSTM, it was observed from the false positive counts of activities done in one location, that they are most frequently confused with each other (e.g. jump's false positive counts were mostly attributed to sit and stand, and likewise for sit - jump and stand, stand - jump and sit). It is hypothesised that the up and/or down motion in one single location for these three actions are similar, resulting in similar signal patterns, and hence led to the prediction errors.

4.1.2 CNN

- The first four CNN models produced similar accuracies, thus proving a CNN based model is able to reliably predict activity for both one- and two-person data in clean and noisy environments with similar accuracy. This is reflected in the combined model results which was on par with the individual models, indicating that it is possible to produce a single end-to-end model which can handle various input image types.
- The recall values broken down by activity show that stand and sit result in the lowest recall values. This means that for these activities the positive cases are getting predicted correctly with least frequency of all five activities. Perhaps this indicates that these activities are getting most confused with each other and this would not be surprising given that these activities are most similar in nature.

- The precision values for the CNN model are relatively similar across the five activities, however note that it is slightly lower for run and sit activities.

4.1.3 Transformers

- The Transformer model reached the highest accuracy compared to the LSTM, ABLSTM, and CNN. The accuracies for all five activities and overall are above 94%.
- Run and walk obtained the highest recall in the Transformer, indicating that the activities that involve changing location are more likely to be correctly identified than those that stay in the same location (sit, stand, and jump). It is hypothesised that the activities in the same location sometimes overlap, leading to incorrect predictions.
- Precision measures the number of true positives out of the total number of predicted positive outcomes. Among all five activities, jump reached the highest precision value of 98%. The high precision of jump might be due to the augmented "time-over-channel" features given by the channel stream of the model. As jump is a repeated action at the same location, the repeated time pattern over the channels can be used to distinguish the activity easily, leading to high precision values.

4.1.4 Comparison of Models

Across all the models investigated, the THAT model gave the highest accuracy results for HAR. This is aligned with theoretical understanding since the THAT model is able to learn both "time-over-channel" and "channel-over-time" features simultaneously.

In general, all models had consistent trends of higher precision than recall values, suggesting that our models are more conservative about their predictions (i.e. models predict positive only if they're confident to do so, which results in higher false negative values and hence lower recall values). Another consistent observation across the models was the confusion among single location activities (i.e. sit, stand and jump confused with each other)) and among two-location activities (i.e. walk and run confused with each other).

4.2 Overall Evaluation

4.2.1 Strengths and Novel Findings

To the best of our knowledge, no previous research has involved HAR of multiple targets. Our project therefore has a significant contribution in that it provides a dataset with both one-person and two-persons data, which can be used for future research, and demonstrates reasonable performance of three different deep learning models on the multi-target problem.

4.2.2 Robustness and Good Practices

We have investigated three different model architectures which provides a degree of robustness against poor performance of one model. Each model was tested on the public dataset and validated against published results before being applied to the JPC3A dataset, which provides confidence in the correctness of the training code.

Additionally, we performed five-fold cross-validation to obtain the reported figures for the final combined models. This allows us to calculate a more accurate and reliable representation of the model's performance, as well as gaining an understanding on the stability of the model. Moreover, it is efficient use of all available data which is important given the limited size of the dataset.

4.2.3 Areas for Improvement

One area of improvement would be to collect more data samples. As mentioned in Section 2, 120 samples were collected for each activity/activity combination in each environment. While this proved sufficient to obtain reasonable model performances, we suspect that more data would enable us to

achieve higher accuracies on our models. Furthermore, a greater number of activities could be investigated beyond the five in this study. For example, activities such as lying down or picking something up could be of interest, particularly whether they can be distinguished from similar activities such as sitting down or standing up. Finally, the number of targets could be increased beyond two to test whether the models are able to distinguish different activities when there are three or more targets present.

4.2.4 Possible Future Works

Besides the extensions to the current study discussed in section 4.2.3, potential future work could also look at multi-modal and/or multi-task research. For example, multi-modal research could explore whether supplementing the CSI signal with information from cameras or wearable sensors improves the accuracy of activity recognition. Multi-task research could attempt to localise the subject of each activity in addition to recognition of the activity.

For the CNN model, a possible future work includes conducting experiments to further understand the effects of image size to model overfitting, and how this varies as the number of targets vary. This would allow us to verify that there does exist a relationship between image size and overfitting, as well as finding the optimal image size.

For the THAT model future work would include working with higher GPU capacity as this was a limiting factor in our research. Greater GPU capacity would allow us to trial larger input lengths in the preprocessing pipeline which could be beneficial because less input information will need to be truncated/lost prior to feeding into the model. Furthermore, similar to the CNN, we could trial different input lengths to see the effect of input lengths and overfitting.

The LSTM/ABLSTM models were also limited by GPU capacity. With higher capacity we could trial larger hidden layer dimensions and/or a greater number of hidden layers, as well as training the ABLSTM with less/no pooling, to determine whether this improves the generalisation capability of the models.

This section covered the assessment of the final models and the team's efforts in ensuring good engineering practices during project execution, and possible future works beyond this current scope.

5 Conclusion

During this project, our team has constructed the first multi-target human activity recognition dataset and defined three different model architectures that were able to achieve reasonable accuracies ranging from 88.6% (LSTM) to 94.8% (THAT) for predicting the correct human activities. The team underwent thorough model design experimentation to obtain the optimal hyperparameters for each model architecture. Overall, it was observed that the THAT model has the highest accuracy when there is no prior knowledge of the number of targets (combined model).

While there are still potential areas of improvement and future work such as more targets and activity types that could be explored, this project's results have provided confidence in the vast potential of multi-target HAR through ML, to improve healthcare, energy and safety applications.

6 Acknowledgement

We would like to express our utmost appreciation and gratitude to our supervisor-in-charge, Professor Julie McCann, for her professional guidance in our project's implementation. We would also like to thank her team for their support. We would like to give special mention to our PhD Mentor, Huang Shuokang, for his help and working closely with us throughout the whole course of the project.

References

[1] LSTM. `https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html`. Accessed: 25 April 2023.

[2] Zhenghua Chen, Le Zhang, Chaoyang Jiang, Zhiguang Cao, and Wei Cui. WiFi CSI Based Passive Human Activity Recognition Using Attention Based BLSTM. *IEEE Transactions on Mobile Computing*, 18(11):2714–2724, 2019.

[3] Charmi Jobanputra, Jatna Bavishi, and Nishant Doshi. Human Activity Recognition: A Survey. *Procedia Computer Science*, 155:698–703, 2019. The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2019),The 14th International Conference on Future Networks and Communications (FNC-2019),The 9th International Conference on Sustainable Energy Information Technology.

[4] Bing Li, Wei Cui, Wei Wang, Le Zhang, Zhenghua Chen, and Min Wu. Two-Stream Convolution Augmented Transformer for Human Activity Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1):286–293, May 2021.

[5] Parisa Fard Moshiri, Mohammad Nabati, Reza Shahbazian, and Seyed Ali Ghorashi. CSI-Based Human Activity Recognition using Convolutional Neural Networks. In *2021 11th International Conference on Computer Engineering and Knowledge (ICCKE)*, pages 7–12, 2021.

[6] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. Whole-Home Gesture Recognition Using Wireless Signals. In *Proceedings of the 19th Annual International Conference on Mobile Computing Networking*, MobiCom '13, page 27–38, New York, NY, USA, 2013. Association for Computing Machinery.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, 2017.

[8] Wei Wang, Alex X. Liu, Muhammad Shahzad, Kang Ling, and Sanglu Lu. Understanding and Modeling of WiFi Signal Based Human Activity Recognition. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, page 65–76, New York, NY, USA, 2015. Association for Computing Machinery.

[9] Siamak Yousefi, Hirokazu Narui, Sankalp Dayal, Stefano Ermon, and Shahrokh Valaee. A Survey on Behavior Recognition Using WiFi Channel State Information. *IEEE Communications Magazine*, 55(10):98–104, 2017.

[10] Xiaolong Zheng, Jiliang Wang, Longfei Shangguan, Zimu Zhou, and Yunhao Liu. Design and Implementation of a CSI-Based Ubiquitous Smoking Detection System. *IEEE/ACM Trans. Netw.*, 25(6):3781–3793, dec 2017.