The Edward S. Rogers Sr. Department of

Electrical and Computer Engineering

University of Toronto

ECE496Y Design Project Course

Group Final Report

Machine Learning for Human Activity Detection on WiFi Signals

Team #: 2021467

Team Members:

Youssef Chmait	youssef.chmait@mail.utoronto.ca
Dylan Araki	dylan.araki@mail.utoronto.ca
Sondre Jaros	s.jaros@mail.utoronto.ca
Ivana Marusic	ivana.marusic@mail.utoronto.ca

Supervisor: Dr. Hojjat Salehinejad

Administrator: John Taglione

Date of Submission: 31 March 2022

Acknowledgements

The team would like to acknowledge Hojjat Salehinejad for his technical assistance and motivation throughout the project, John Tagilone for his role in helping the group improve its communication, Konan Phang for running the course, the developers of Widar 3.0 for developing the B.V.P. dataset, the developers of Minirocket for providing fast time-series classification, and all students and faculty at the Edward S. Rogers Sr. Department of Electrical and Computer Engineering for an enjoyable university experience.

Executive Summary

The team aims to develop a new wireless system using machine-learning algorithms for detecting human gestures performed in indoor environments that is both more accurate and computationally cheaper than existing state-of-the-art approaches at classifying a given gesture. The team's scope extends only to processing and classifying existing data used by the developers of Widar 3.0, the current state-of-the-art model. This data is known as a body-coordinate velocity profile (B.V.P.), which is a processed form of channel-state information (C.S.I.). The processing used to generate it aims to eliminate the effects created by information irrelevant to the gesture being performed, such as the person's orientation relative to the transceivers and receivers and the person's non-moving body parts, leaving the distribution of signal power over the velocity components. The team's approach uses random, one-dimensional convolutional kernels to extract features that a ridge-regression classifier then proceeds to use to classify a time-series of B.V.P., which has been shown to achieve high accuracy and greatly reduced time complexity on time-series datasets. In testing, the team determined that the model meets all the project goals and requirements for the computational complexity, and the majority of project goals for classification accuracy. The significant improvement in computational complexity, with only a minimal reduction in accuracy, is a beneficial tradeoff when this software is deployed in the field with limited hardware and memory constraints.

Table of Contents

1.0 Introduction

This report outlines results on applying random convolutional kernels to body-coordinate velocity profile (B.V.P.) data—a processed form of time-series channel-state information (C.S.I.) from Wi-Fi transceivers and receivers—for the purpose of classifying human hand gestures, concluding with recommendations for areas of further exploration.

1.1 Background and Motivation

Many applications ranging from smart homes to medical diagnoses require accurate information about people's movement patterns, yet it is not always viable for subjects to wear cumbersome equipment or be filmed. To avoid these problems, researchers have begun experimenting with Wi-Fi-signal data, which is abundant in indoor environments, as a non-intrusive way to collect gesture information [1, 2]. This presents an opportunity to employ machine-learning techniques to determine the correspondence between various interference patterns in Wi-Fi transceivers and receivers and recognizable human gestures.

Previous attempts relying on Wi-Fi channel-state information (C.S.I.)—which encodes how signals propagate between transceivers and receivers allowing them to identify disturbance patterns created by people performing gestures—included limitations inherent in the use of C.S.I. [1, 2]. For instance, in Figure 1, although the two individuals are performing the same gesture, their relative orientations to the transceivers and receivers leads to the waveforms hitting the receivers at different angles. Also, interferences in the wave propagations are created by their bodies' mere presences, despite only their arms performing the gestures. To address this problem, researchers at Tsinghua University developed a body-coordinate velocity profile (B.V.P.)—a processed form of C.S.I. that filters out irrelevant information (described further in section 2.1). Using this data, the researchers combined a convolutional and recurrent neural network (C.N.N. and R.N.N.) to become the state of the art for classifying microgestures (ones in which the person's feet remain stationary) [1].

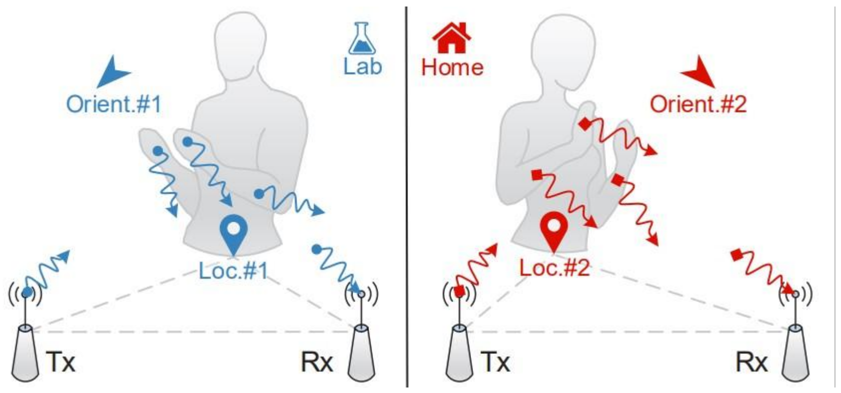


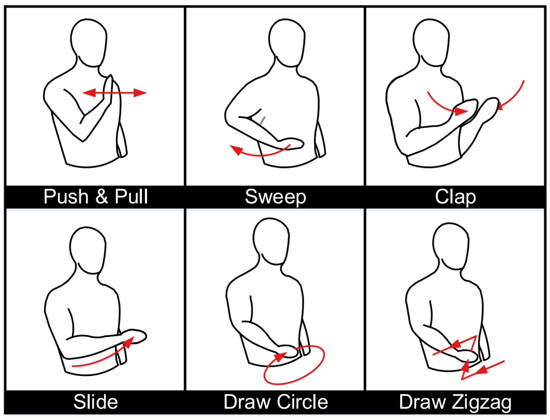
Figure 1 [1]

While B.V.P. proved to be more successful than existing approaches relying solely on C.S.I., its multi-layered model includes on the order of 10^5 trainable parameters, which causes high training times. However, because of recent research showing models simpler than deep neural networks, and thus faster, can achieve similar accuracies on time-series data, it was decided to research the effects of alternative models on B.V.P [3].

1.2 Project Goals and Requirements

The following requirements were defined:

Requirement	Justification
All testing will be done on the same B.V.P. dataset used in the development of Widar 3.0.	The purpose of the project is to assess alternative approaches to Widar 3.0, the state of the art, using B.V.P.
Models will classify the following six gestures: Push/Pull, Sweep, Clap, Slide, Draw ZigZag, Draw-Circle (see Figure 2)	These were the six gestures focused on by the Widar paper. So for comparison, the same set of gestures is used.
The model should classify gestures performed in both the room it was trained on (in-domain) and rooms it was not trained on (cross-domain).	Applications of gesture recognition could require either case.



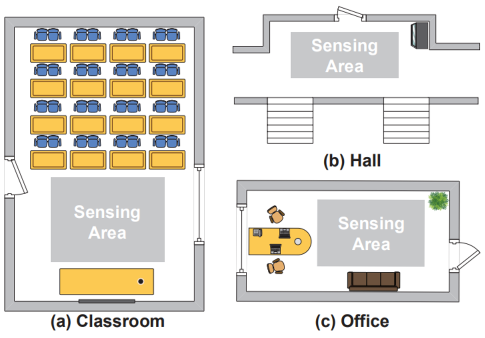


Figure 2 [1] **Figure 3 [1]**

The following project goals were defined:

Goal	Metric
Achieve higher average accuracy than Widar 3.0 for in-domain tests for each room (see Figure 3 for rooms tested in).	Train both models using a random 90% sample of the room's data and then measure the percentage of correct classifications on the remaining 10% of the data. Repeat this process five times and then average both models' accuracy on the room.
Achieve higher average accuracy than Widar 3.0 for cross-domain tests when training on Room 1.	Train a model on all of the data on Room 1 (a classroom, see Figure 3) and test on all of the data on another room. Repeat this process three times and then average both models' accuracy on the room.
Achieve a higher average accuracy than Widar 3.0 on tests using all the data available across all three rooms.	Train both models using a random 90% sample of all the data and then measure the percentage of correct classifications on the remaining 10% of the data. Repeat this process three times and then average both models' accuracy on the room.
Achieve higher accuracy on each test on at least half the gestures.	For every test, measure the average accuracy by gesture and determine whether at least three of the gestures have a higher average accuracy.
Use fewer trainable parameters than Widar 3.0.	Measure the number of trainable parameters for both models.
Require lower training time than Widar 3.0.	Measure the amount of time required to train both models.
Require lower inferring time than Widar	Measure the amount of time required to

3.0.	infer for both models.

2.0 MiniRocket Model

Walkthrough of System Diagram

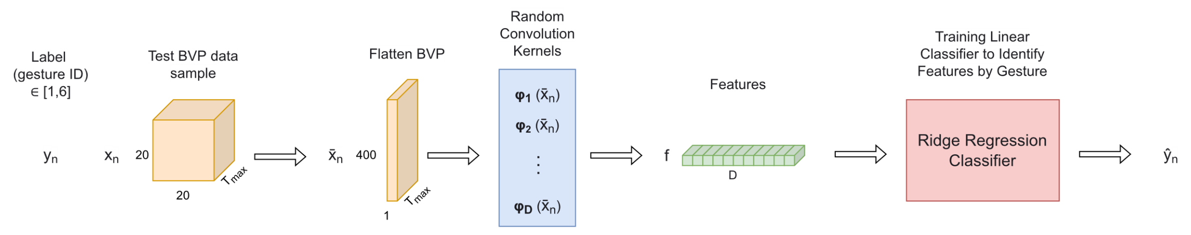


Figure 4

The B.V.P. input data consists of six different gestures. The model uses the B.V.P. data of the shape of $20x20xT_{max}$, where T_{max} is the number of time series snapshots there are of the gesture and the 20x20 is the representation of the movement where the user is at the origin. The data gets vectorized to be of the shape $400xT_{max}$.

The model then takes this flattened B.V.P. data and runs it through the random convolution kernels. These kernels are classified as random because they are assigned random weights. The model used, which is called Minirocket for Minimally Random Convolution Kernel Transform, is a single-layer convolutional neural network with randomly selected kernel weights. This model has the benefits of being faster than the previous state of the art for training time-series data while still retaining a good accuracy. It has been shown that the random weights of the kernels greatly reduce the time complexity of the transform without taking away from the accuracy of the results. Previous methods for time-series data require multiple layers of learning to achieve the team’s levels of accuracy. This method is also more efficient for smaller datasets, like those of the B.V.P. datasets used. From these kernels features are extracted, which are then used to train a linear classifier to learn the mapping from features to labels/classes.

The ridge-regression classifier is used to train the data as it is the fastest for the dataset size. This specific classifier is significantly faster than other linear classifiers on smaller datasets.

2.1 Input Signals

This section gives a brief overview of how the dataset was generated.

2.1.1 Collection of Data

To collect the data, individuals of various heights and weights were placed in locations around three different rooms (see Figure 3) and performed the relevant gestures with their bodies pointed at various angles relative to the floor (see the location markers and orientation angles in Figure 5). Transceivers would emit Wi-Fi signals that would interact with various receivers, allowing for the collection of channel-state information describing the waveforms generated as the subjects performed the gestures. This data is then preprocessed to remove amplitude noise and phase offsets [1].

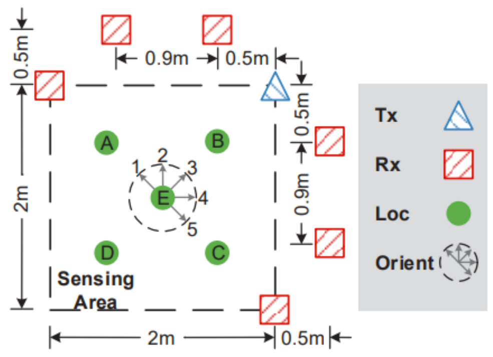


Figure 5 [1]

2.1.2 Conversion of C.S.I. to B.V.P.

The B.V.P. attempts to capture the distribution of signal power over velocity components over a plane perpendicular to the ground and parallel to the person's height (the local-body coordinates). Thus, any effects created by a person's relative position to the transceivers and receivers will be filtered out so that only information about how the person's body parts move will be included [1].

The C.S.I. is then transformed into an intermediate state known as the Doppler-Frequency State (D.F.S.). The generation of the D.F.S. filters out band noises and static offsets by performing conjugate multiplication on the C.S.I. from two receivers. To this product, a

Fourier Transform is then applied, giving the power distribution over time and Doppler frequency domains in a matrix of size FxM, where F is the number of sampling points in the frequency domain and M is the number of transceiver links—this outcome is the D.F.S. for each timeshot [1].

The D.F.S. is then quantized into a NxN matrix (N=20 in the B.V.P.) so that the local-body coordinates are at the origin, where N is the number of possible values of velocity components decomposed along each axis of the local-body coordinates. By projecting the D.F.S. velocity values onto the local-body coordinates, the information will be independent on the location of the transceivers and receivers. Static components without D.F.S. information are then filtered out to prevent interferences created from non-moving body parts or open air from affecting the output. The corresponding output is the B.V.P. in the size of $NxNxT_{Max}$, where T_{Max} is the maximum number of timeshots for a given sample, which is then normalized [1].

2.1.3 Data Processing

When generating the B.V.P. data, padding is added along the temporal dimension to match the maximum number of snapshots of any of the sample. Next, the B.V.P. data is then rearranged into Sktime format—the type used for Minirocket's feature extraction—which requires the measurements being the rows and the time snapshots being the columns. Thus, the NxN B.V.P. is flattened to be a length-N^2 vector for all time snapshots, and then the transpose is taken for each time-series sample.

2.2 Kernels

Convolutional kernels work to extract features from an input. The team's model uses a large number of random convolutional kernels rather than the usual learned convolutional kernels to extract features since it has been shown in Minirocket that random convolutional kernels capture the features needed for time series classification.

These kernels in the transform all have a length of nine, in which the weights are fixed from the list of {-1,2}, which means that the kernels are all two-valued kernels. The biases are determined from the convolution output of the kernels and the input data. From these kernels the transform computes the proportion of positive values, P.P.V., pooling which is what is used to extract features from the kernels. The dilation is used to spread a kernel over the input data, where a kernel is convolved over every fixed number of input elements. Every kernel gets a fixed set of dilations, and each integer dilation determines the features computed per dilation. Padding is alternated for each kernel/dilation combination so half the combinations use padding, and half do not, and standard zero padding was used.

2.3 Transform

Once the kernels are generated, time-series samples (of size Tx400) are reduced to one dimension of 10,000 features. To do this, a feature map is generated by applying a sliding dot product between the generated kernel and the input data, as is standard in convolutional transformations. These values are then pooled using the proportion of positive values (the number of samples in which the bias is less than the weighted sum of the inputs) [3]. The primary difference from normal convolutional networks being Minirocket's reliance on a large number of randomly generated convolutional kernels to extract the features. Thus, the convolutional layer is broad rather than deep and is applied to a flattened form of the B.V.P. for each time slot.

2.4 Classifier

Minirocket is a transformer which produces features. These features are used to train a linear classifier, and the team's model can be run with any classifier. The has chosen the ridge-regression classifier, a linear classifier that, for smaller datasets like the B.V.P., is very fast while also achieving high classification accuracy. The ridge-regression classifier is fast on smaller datasets because it uses a generalized cross validation to determine appropriate regularization. The classifier uses eigen decomposition, which works well for datasets with fewer training samples than features, like the B.V.P. datasets.

2.5 Complexity

The transform has a linear complexity of $O(k \cdot n \cdot l_{input})$, where k is the number of kernels, n is the number of training examples and l_{input} is the time series length. This means that the complexity is proportional to the number of kernels, which is fixed at 10,000. The ridge-regression classifier has a complexity of $O(N^2 \cdot D)$ where N is the number of training samples and D is the number of features when $N < D$ and $O(N \cdot D^2)$, when $N > D$.

3.0 Experiments

3.1 Data Processing

The experiments were conducted directly on the Body-Coordinate Velocity Profile data provided by Widar [1]. This data was sorted by physical location (Room 1, Room 2, or Room3) as designated by the Widar README [1], and then uploaded to a Google Drive [3] for use in Google Colaboratory. Samples were then extracted from the BVP dataset corresponding to the six primary gestures identified in [1]: push, sweep, clap, slide, draw-zigzag, and draw circle. Each sample was loaded in as a NxNxT tensor, where N represents the size of each spatial dimension of the BVP series, and T is the number of snapshots in the BVP series. The samples were normalized and zero-padded along the time axis to the maximum number of snapshots (T_{max}).

3.2 Setup

The model was implemented in Python in Google Colaboratory [4]. For in-domain testing, a 90% training/10% testing split was used within each room. For cross-domain testing, the model was trained on the entirety of room 1 and tested on both room 2 and room 3. An "all-domain" test was also conducted in which samples were randomly selected from the entire dataset for training and testing with a 90% training/10% testing split. In each case, the average result of 5 independent runs was taken.

3.3 Classification Performance

Table 1 and Table 2 compare the in-domain and cross-domain of Widar D.N.N. and the team's model using Minirocket. To analyze in detail the differences in performance between gestures, confusion matrices are provided in Table 4(a) and Table 4(b).
The average in-domain and cross-domain accuracy of the MiniRocket model is slightly lower than that of Widar D.N.N. when trained on Room 1 but higher when trained on the smaller datasets of Rooms 2 and 3. Thus, Minirocket offers better relative performance when the number of training samples is limited.

3.3.1 Gesture-Specific Classification Performance

For both Widar 3.0 and the team's Minirocket-based model, accuracy is generally high for Push/Pull, Sweep, Clap, and Slide, but generally poor for Draw-Zigzag and Draw-O. As the transceivers used in collection of CSI data are placed at the same height, they do not accurately capture vertical-velocity components. The result is high accuracy on gestures involving mostly horizontal motion but poor accuracy on gestures that involve significant vertical motion. This discrepancy in accuracies likely reflects a weakness in the collection and processing of the B.V.P. data itself, rather than inherent weaknesses in using convolutional networks on motion data.

Relative to Widar 3.0, the team's model achieves a higher performance in Push/Pull and Sweep. Relative to Minirocket, Widar achieves a higher performance in Slide and, although still poor, higher performance in Draw-Zigzag and Draw-O. It may be recommended that, if deployed in field applications in its current state, both Widar and Minirocket should be used on simpler, more horizontal motions.

3.4 Computational Complexity

Although the team's model is not conclusively more accurate than Widar 3.0, it offers vastly improved algorithmic complexity. As Google Colaboratory does not provide a consistent hardware resource (a different CPU is assigned each runtime), the recorded training and inference times give only a rough estimate of the relative time requirements of each method. However, as seen in Table 1, MiniRocket was found to execute between 4 to 10 times faster in both training and inference time per sample. This difference is significant enough to conclusively show that, despite any variations in hardware between runtimes, the use of Minirocket offers a significant improvement in time complexity.

3.5 Results

Method	Push/Pull	Sweep	Clap	Slide	Draw-Zigzag	Draw-O	Average:	Training Time Per Sample (sec)	Inference Time Per Sample (sec)
Widar DNN	**Room: (1)0.85 (2)0.89 (3)0.94**	**Room: (1)0.82 (2)0.96 (3)0.95**	**Room: (1)0.99 (2)0.98 (3)0.94**	**Room: (1)0.84 (2)0.24 (3)0.90**	**Room: (1)0.63 (2)0.31 (3)0.32**	**Room: (1)0.47 (2)0.37 (3)N/A**	**Room: (1)0.80 (2)0.69 (3)0.80**	**Room: (1)0.0406 (2)0.0460 (3)0.0393**	**Room: (1)6.51e-5 (2)1.15e-4 (3)1.85e-4**
MiniRocket	**Room: (1)0.97 (2)0.99 (3)0.98**	**Room: (1)0.89 (2)0.98 (3)0.98**	**Room: (1)1.0 (2)1.0 (3)1.0**	**Room: (1)0.70 (2)0.40 (3)0.79**	**Room: (1)0.48 (2)0.36 (3)0.70**	**Room: (1)0.15 (2)0.40 (3)N/A**	**Room: (1)0.75 (2)0.75 (3)0.88**	**Room: (1)0.01 (2)0.005 (3)0.002**	**Room: (1)1.45e-5 (2)3.5e-5 (3)8.29e-5**

Table 1: In-Domain Results per Room

Method	Push/Pull	Sweep	Clap	Slide	Draw-Zigzag	Draw-O	Average:
Widar DNN	**Room: (2)0.85 (3)0.84**	**Room: (2)0.80 (3)0.77**	**Room: (2)0.99 (3)0.99**	**Room: (2)0.66 (3)0.83**	**Room: (2)0.40 (3)0.74**	**Room: (2)0.36 (3)N/A**	**Room: (2)0.73 (3)0.83**
MiniRocket	**Room: (2)0.96 (3)0.95**	**Room: (2)0.90 (3)0.83**	**Room: (2)1.0 (3)1.0**	**Room: (2)0.62 (3)0.94**	**Room: (2)0.24 (3)0.49**	**Room: (2)0.10 (3)N/A**	**Room: (2)0.72 (3)0.84**

Table 2: Cross-Domain Results (trained on Room 1)

3.5.1 Confusion Matrices

	Push/Pull	Sweep	Clap	Slide	Draw-Zigzag	Draw-O
Push/Pull	**0.88**	0	0.01	0.11	0	0
Sweep	0	**0.85**	0	0	0.14	0.01
Clap	0	0	**0.99**	0	0	0.01
Slide	0.15	0	0	**0.72**	0.10	0.03
Draw-Zigzag	0	0.23	0	0.09	**0.59**	0.09
Draw-O	0	0.04	0	0.07	0.43	**0.46**

(a) Widar DNN [1]

	Push/Pull	Sweep	Clap	Slide	Draw-Zigzag	Draw-O
Push/Pull	**0.98**	0	0.01	0.01	0	0
Sweep	0	**0.95**	0	0.01	0.04	0
Clap	0	0	**1.0**	0	0	0

Slide	0.25	0.01	0	**0.67**	0.07	0
Draw-Zigzag	0	0.32	0	0.25	**0.41**	0
Draw-O	0	0.08	0.01	0.26	0.36	**0.20**

(b) MiniRocket

Table 3: Confusion Matrices for Combined Data (All-Domain)

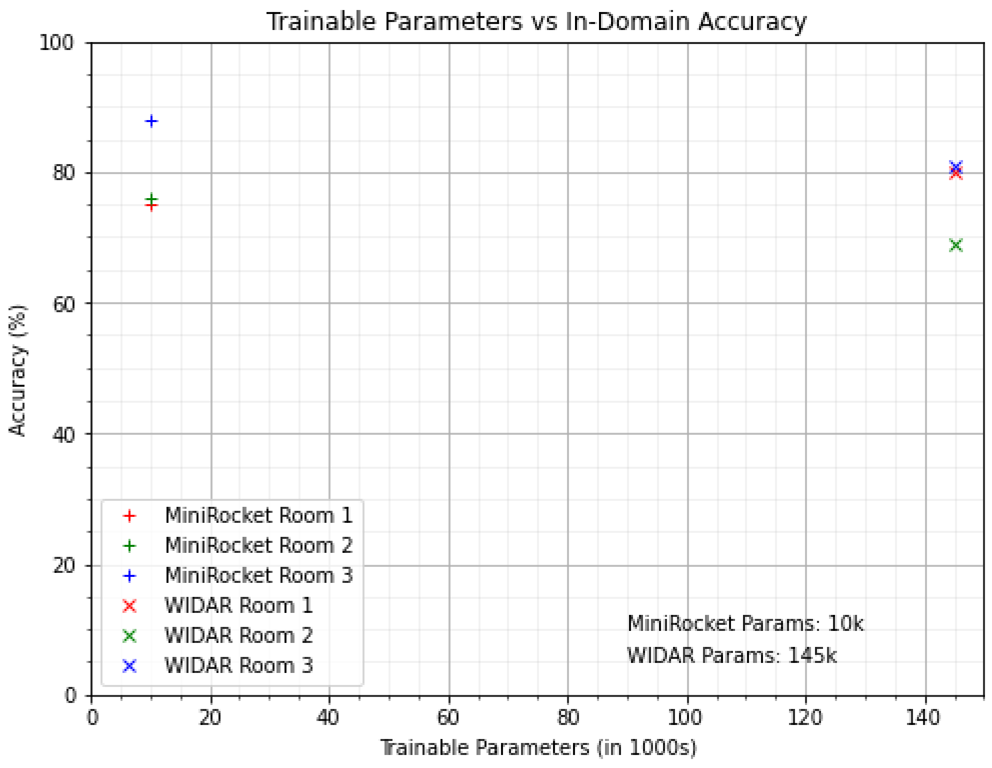


Figure 1: Trainable Parameters vs In-Domain Accuracy

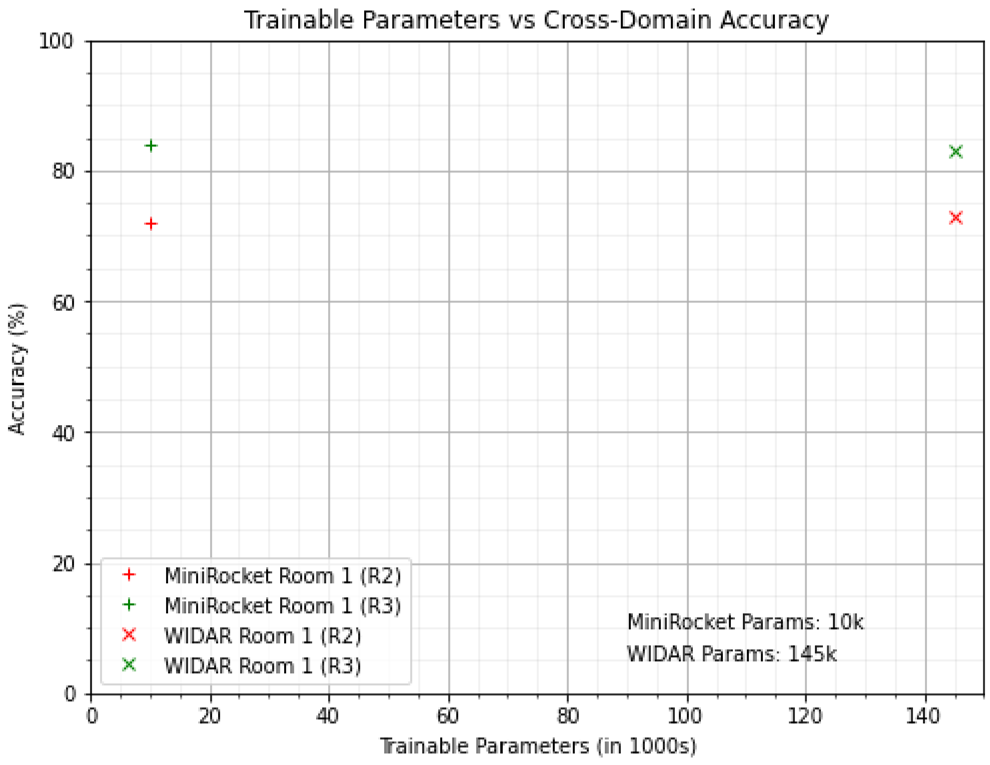


Figure 2: Trainable Parameters vs Cross-Domain Accuracy

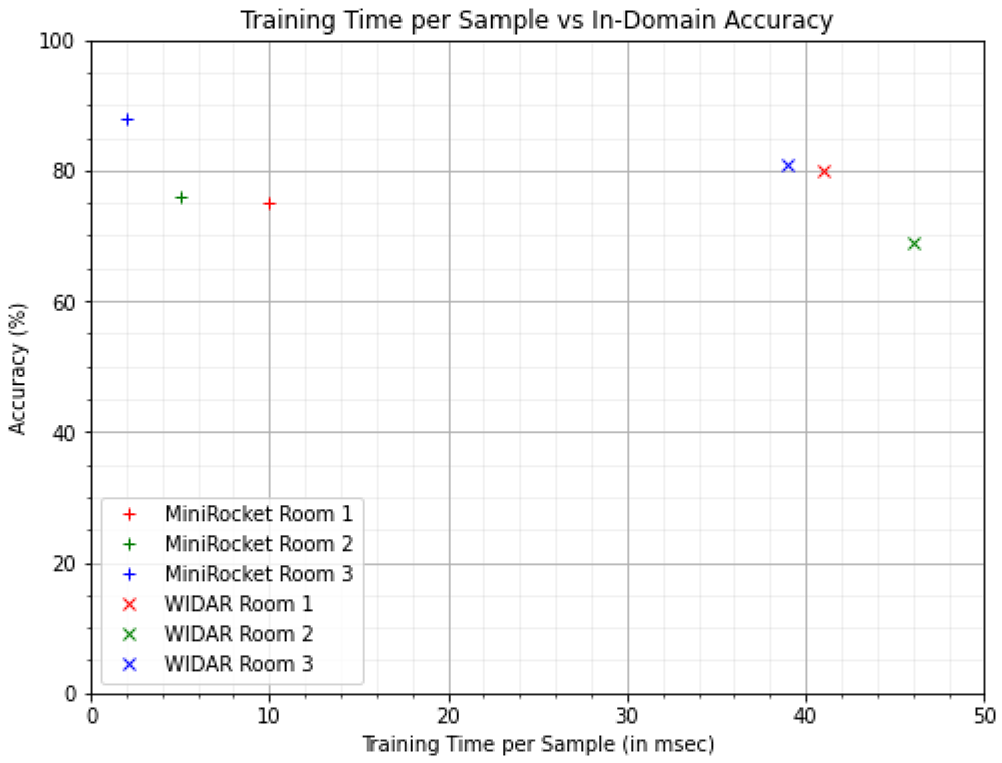


Figure 3: Training Time per Sample vs In-Domain Accuracy

4.0 Summary and Conclusion

The team has successfully implemented a model that accurately classifies human gestures detected by the disruption of Wi-Fi signals. Overall, the team's model performs best on smaller datasets in accuracy, only falling slightly behind in accuracy with large datasets, and in computational complexity, the team has surpassed Widar 3.0 in all test samples.

In testing, the team determined that the model meets all the project goals and requirements for the computational complexity, and the majority of project goals for classification accuracy. The significant improvement in computational complexity, with only a minimal reduction in accuracy, is a beneficial tradeoff when this software is deployed in the field with limited hardware and memory constraints.

The work done in this project has several applications. Current motion detecting systems generally require extra installation of motion detectors or cameras or even on-person devices. As Wi-Fi is readily available in most indoor areas, Wi-Fi human activity recognition can offer an accessible solution for fall detection of the elderly or smart homes.

Further research should focus on improving the extraction of vertical velocity components from CSI data, which would allow for higher recognition accuracy for gestures with more significant trajectories in the vertical direction.

6.0 References

[1] Zheng Yang, Yi Zhang, Guidong Zhang, Yue Zheng, December 26, 2020, "Widar 3.0: WiFi-based Activity Recognition Dataset", IEEE Dataport, doi: https://dx.doi.org/10.21227/7znf-qp86. [2] S. Yousefi, H. Narui, S. Dayal, S. Ermon and S.
[2]Valaee, "A Survey on Behavior Recognition Using WiFi Channel State Information," in IEEE Communications Magazine, vol. 55, no. 10, pp. 98-104, Oct. 2017, doi: 10.1109/MCOM.2017.1700082.
[3]Dempster A, Petitjean F, Webb G (2020) ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min Knowl Disc 34:1454–1495