Indoor Activity Recognition with mmWave Radar Sensor

Mingdian Liu
Ge Luo
Shuhan Yang

Team 8

April 25, 2021

Final Project for
HCI575: Computational Perception

Iowa State University
Spring 2021

ABSTRACT

Nowadays, world population aging, which requires heavy investment of time and manpower in daily care, is having a heavy impact on society and economic of human beings. To caregivers, injury or death caused by fall-down and declined self-dependence due to Alzheimer's disease are two main issues to struggle on. Herein, we devise a human activity recognition (HAR) system which is potential to handle these issues. In contrast to traditional system using RGB camera, the indoor activity recognition system with mmWave Radar offers an service with low privacy leakage risk without costing any recognition accuracy. Taking the data of daily indoor activities as input, four deep learning models are trained and compared to build up an optimized activity classifier. The averaged accuracy of best model in test data set reaches 91.87%, which shows its full potential for practical deployment. A simplistic user interface (UI) is also designed to demonstrate the functionality of well-trained model. To implement distributed deployment, we connect Raspberry Pi to AWS IoT for real-time data collection and processing respectively. A probability threshold of fall-down action is set by AWS Lambda to trigger an alarming and call for first-aid in time. We hope the proposed system could provide an accurate recognition service which further enable the remote monitoring for elderly people and reduce labour for caregivers.

1 Introduction

A rapid increase in the percentage of elderly people over the past few years has been a cause of huge pressure on the economic and social life of all countries[1]. Expectations show that 20% people will be older than 65 by 2030. Active research is being carried out to leverage the benefits of information and communication technologies that enable them to live independently and promote a sense of overall well-being. Among these researches, the serious consequence of **falling down** and **Alzheimer's disease** of elderly people has raised the concern of many investigators (see Figure 1).

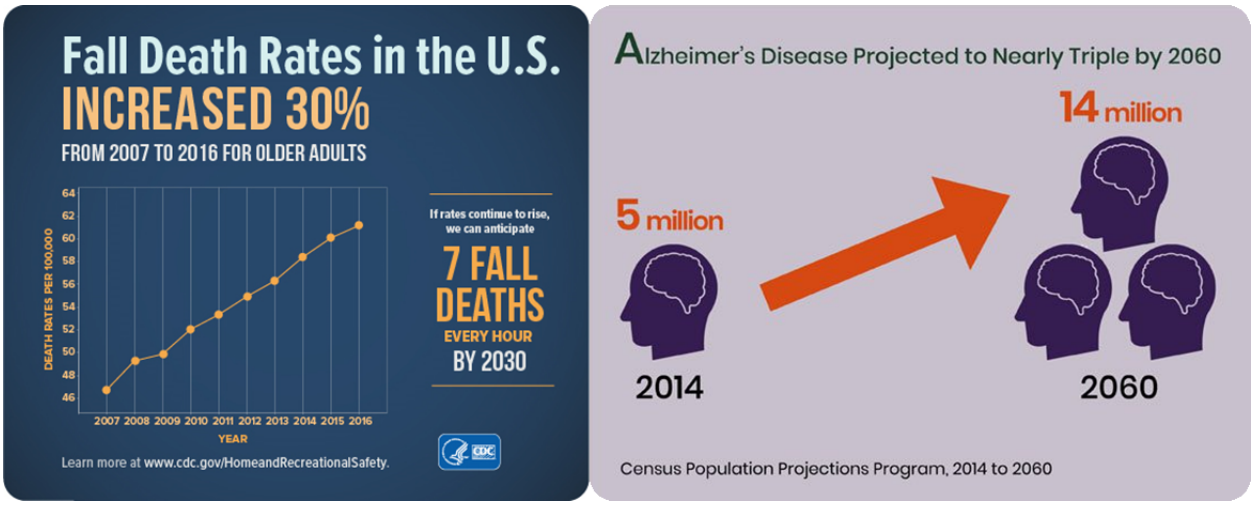


Figure 1: Fall-down and Alzheimer's disease are two main issues both elderly people and their caregivers are struggling on.

According to the report released by Centers for Disease Control and Prevention (CDC)[2], more than one out of four **older people falls** each year, but less than half tell their doctor. One out of five falls does cause a serious injury such as a broken bone or a head injury. These injuries can make it hard for a person to get around, do everyday activities, or live on their own. More seriously, it is expected that 7 fall deaths every hour by 2030 caused by not immediate emergency aid. Therefore, a real-time system of detecting fall-down and sending alarms promptly is in demand to reduce death and improve the living quality for senior people.

On the other hand, the statistics reveal that about 3% of people aged 65 to 74, 19% of people aged 74 to 84 and nearly half of people more than 84 will have **dementia**[3].People who are diagnosed with dementia often struggle to engage in discussions, to remember scheduled events, and to determine appropriate words for expressing situations. With the progress of dementia, they forget names, repeat words, face orientation problems, become confused about date and time, and wander around during the night times, which evokes much burden to caregivers. In result, a 24/7 monitoring system for elderly people would substantially facilitate caregivers' responding and reduce their manpower.

Some review paper indicates that **Human activity recognition (HAR)** has the potential to be employed to the smart-home facility and solve the above two issues. For a typical HAR system showed in Figure 2, it includes data collection, data pre-processing, training, and activity recognition in order. The functionalities of HAR system comprises people identification, activity recogni-

tion, and tracking. HAR is profusely used in emerging applications and services in smart spaces, such as personalized heating and cooling, security management, efficiency monitoring, natural light adjustment, background music selection, etc. However, employing HAR system for elderly care is yet well investigated. From the perspective of caregivers, the activity recognition system should be implemented in real time and has the ability to notify caregiver once any abnormality takes place. This daily activities record in the system can be utilized for habit checklist and early diagnosis of some chronic diseases.

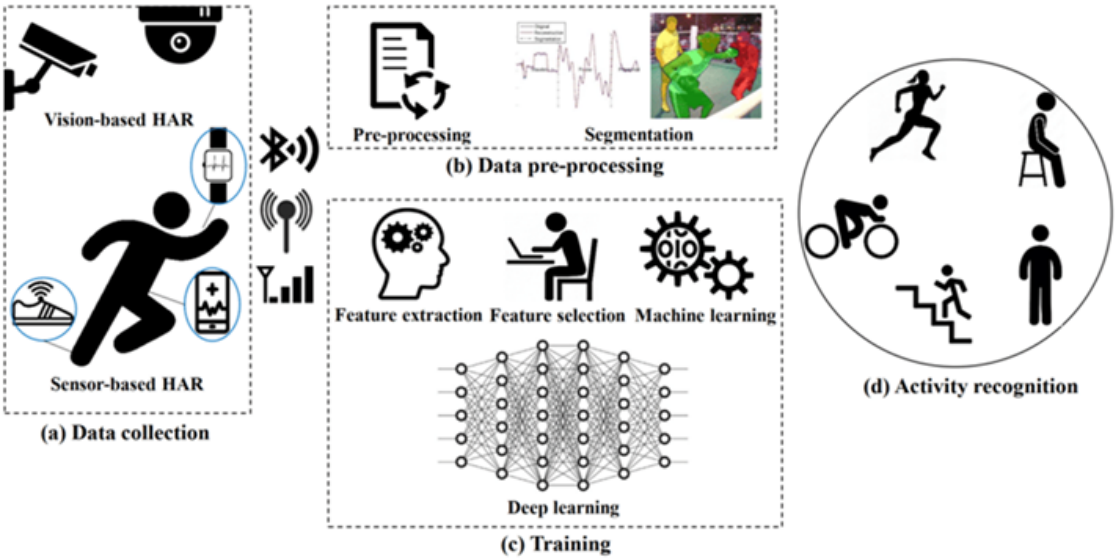


Figure 2: A typical framework of HAR system

This project predominantly focuses on the utilization of activity recognition system for elderly people monitoring by radar sensor. The IWR6843 mmWave sensor is deployed in an experimental scenario to collect the indoor activities including workout, walk, fall down, stand up, and sit down. Four deep learning models (CNN+Transformer, Multi-MLPs+ Transformer, Multi-CNNs+ Transformer, and BiLSTM) are devised and compared for the activities recognition accuracy. The best averaged accuracy on test data set reaches 91.87%, which implies the full potential of this system in the application of assisting elderly care. A deployment roadmap is also proposed. In the roadmap, the point cloud data from radar sensor is transmitted to AWS IoT via Raspberry Pi. The pre-training activities classification model is preset on AWS Lambda. As long as the predicted fall-down probability exceeeds the threshold value, a immediate alarming would notify caregiver and emergence center and call for first-aid. The detected indoor activities history can also be checked by end-users via a WebApp. In conclusion, the system we develop is innovative and practical. We hope this system would help caregiver of elderly people reduce time and manpower cost in the future.

2 Related Work

Figure 3 shows multiple sensor choices for the application of indoor activity recognition. Generally, we can categorize them in to wearable sensor for device-based solution and unwearable

sensor for device-free solution. Most of them have been well investigated by researchers for a long time. Herein, by listing related works on indoor activity recognition, we compare the advantages and disadvantages of device-based solution and five types of unwearable sensors (i.e. floor sensor, RGB camera, depth camera, WiFi CSI, and radar sensor). To highlight the practicality of radar sensor, Table 1summarizes the comparison details.

2.1 Device-based method

Some device-based methods, which requires the carried token, such as ID/swipe cards, active badge, smartphone, smartwatch has been deployed in smart spaces to identify users' activities by the uniqueness of personal devices. For instance, a systematic solution combining smartphone and smartwatch is proposed for indoor activities monitoring. The whole system can be run in real time and reaches as high as 98% for activities classification. However, the inseparable relationship between user and their identifying device narrows down the function of these technologies to some limited scenarios.

2.2 Device-free method

In order to unfreeze the restriction of device-based methods, device-free methods have been proposed and are increasingly being adopted for activity recognition. Vision based techniques (e.g., cameras) are widely used methods in this category and outperform other sensors on the condition of a clear, frontal view of the activity. However, due to the customer's concern on intrusive method, camera have a low acceptance in domestic and commercial settings[4]. On the other hand, electromagnetic wave based methods are less intrusive and have also ability of collecting enough information for activity recognition. In particular, it has been proposed to utilize the variations in ambient WiFi signals to recognize people's indoor activities[5, 6]. Despite the general deployment of WiFi module, such method is restricted to the narrow area between the separate transmitter and receiver. Having a similar principle to WiFi module, the mmWave radar is a compact chip combining receiver and transmitter so as to only demand a single device for deployment. More essentially, the CSI-based WiFi signal is extracted from physical layer of WiFi module, which is incapable of concurrently recognizing activities of multiple people in the same scene. More importantly, the mmWave sensor can collect enough information for accurate activity recognition meanwhile keep the collected data not as rich as camera which has the privacy risk.

Another two sensor options of activity recognition are LiDAR and depth cameras which have also been investigated for a long time[7]. However, LiDAR is not affordable for house usage and depth cameras is only limited to a small sensing range due to the short wavelength it uses. Meanwhile, as an optical based sensor, they also make user have the same concerns on privacy leakage as with conventional cameras.

2.3 Radar sensor

After the complete comparison, the mmWave sensor outperforms other potential sensor choices due to its high accuracy, least concern on privacy leakage, capability for multiple people detection and tracking, as well as easy and environment-independent deployment. The mmWave radar sensor has been introduced for multiple-people tracking and human identification[8]. The vexolization method used in this paper inspired our work. The main difference between reference[8] and our

Figure 3: Typical sensors for HAR system: floor sensor, RGB camera, Depth camera, WiFi CSI, and radar sensor.

	Floor Sensor	RGB Camera	Depth camera	Wifi CSI	Radar Sensor
Identification accuracy	Moderate	Very high	High	Moderate	Moderate
Recognition accuracy	Moderate	Very high	High	Moderate	High
Tracking	Yes	Yes	Yes	Yes	Yes
Environment independent	No	Yes	Yes	No	Yes
Privacy concerns	None	High	Medium	Low	Low
Ease of deployment	Very difficult	Easy	Easy	Difficult	Moderate

Table 1: A comparison between different sensor choices

work is that we focus on indoor activity recognition. Another paper also show the application of mmWave sensor in workout activity recognition [9]. However, the fall detection is not investigated in this paper. Also, the limited detection range of radar sensor used in this paper also restrict the improvement of classification accuracy.

In this project, we will explore the utilization of mmWave sensor for indoor activity recognition especially fall detection for elderly people.

3 Data collection

3.1 Equipment

The equipment we used is a commercial mmWave Radar sensor from the Texas Instrument. The model name is IWR8643BOOST[10] which works at a frequency range from 76-GHz to 81-GHz. The sensor has on-board antenna hence is easy for indoor deployment. The approximate effective range for the sensor is from 6 to 10 meters which can cover a large area for a room.

The sensor transmits millimeter waves, and receives the reflections. The sensor supports two kinds of output format.

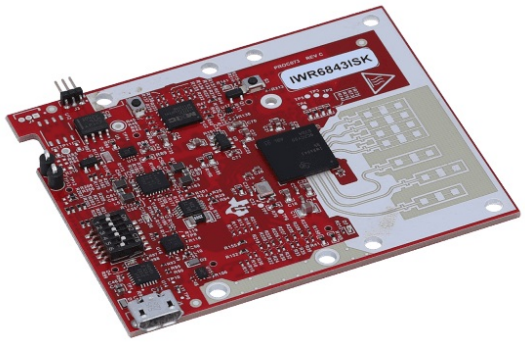


Figure 4: IWR8643 packaged mmWave sensor with DSP and MCU evaluation module

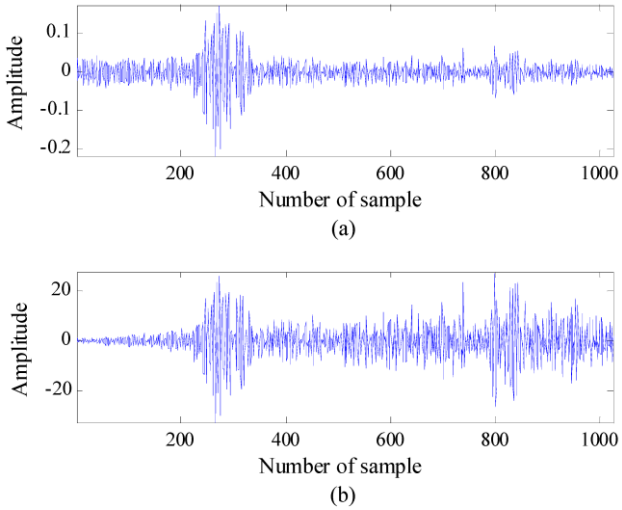


Figure 5: Example raw data

1. Raw data: As shown in Figure 5. Very much like a microphone which captures the sound signal, the raw data produced by the sensor is the mmWave signal which shows the amplitude along the time captured by each channel. The sample rate of the device is 625 ksps.

2. 3D point cloud: the point cloud data can be obtained using the onboard DSP and MCU evaluation module. As can be seen from Figure 6, in the data (x, y, z) coordinates of points represents the objects detected. In addition, the each point also have two extra attributes: velocity and intensity which often give information about the object's type. The frame rate given by the device output is 30 fps.

The raw data produce larger amount of data per second and contains much more noise. Thus, we decided to use the 3D point cloud data for our system.

3.2 Dataset

Firstly, we used a public dataset called MMActvity[9] at the beginning for the algorithm development. Then the algorithm are migrated using the dataset collected by ourselves. Here is the specification for the collection process: TI IWR6843 mmWave sensor is mounted on a stable tri-

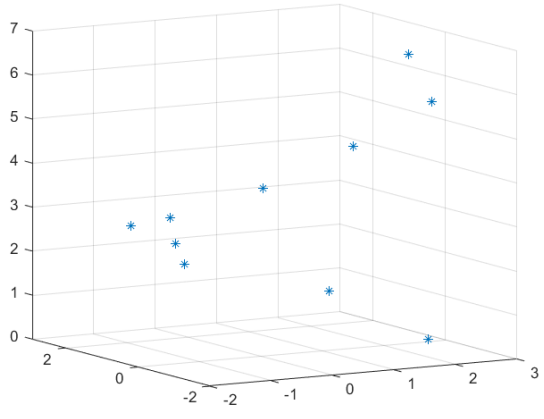


Figure 6: Example 3D point cloud

pod in a height of 1.6 m. The antenna array is 15 degree tilted to get directed to the body of experiment participants. Different activities including workout, walk, fall down, stand up, and sit down are performed in front of the radar sensor in a distance of 3 m. Each recording consists 30 seconds of one user performing the same activity, the total length of the dataset is 46 minutes. Each activity has about the same total length.

Finally the dataset are split for training purpose, we have 2484 samples in the training set, 828 samples in the validation set and 828 samples in the test set.

4 Proposed approach

Below we show the pipeline of a standard supervised learning techniques used to achieve the indoor activity recognition.

4.1 Architecture overview

The architecture of our proposed approach can be seen from Figure 7. The installed mmWave Sensor outputs point cloud data. These data are pre-processed and the send to classifier as inputs. In the end, an activity label is classified to represent the activity currently detected. We will describe the details for each of the stage in the following chapter.

4.2 Data Preprocessing

The point cloud data are preprocessed[1] (See figure 8) so it can be used as input for classifiers.

The points in each data frame are first sorted by its z-coordinate and then we apply zero-padding to each data frame to a fixed length of 42 points which is selected based on maximum number of points that can appear in a frame in the dataset. Finally, a sliding window of 2s (60 frames) is chose to contain the information for movement in an activity. The overlap between two windows is set

[1]In the project proposal, we mentioned voxelization and clusering as two possible preprocessing methods. However, they are not presented in the final report due to the limited time, and are thus not implemented yet.

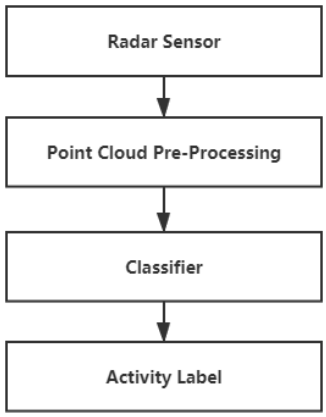


Figure 7: Architecture

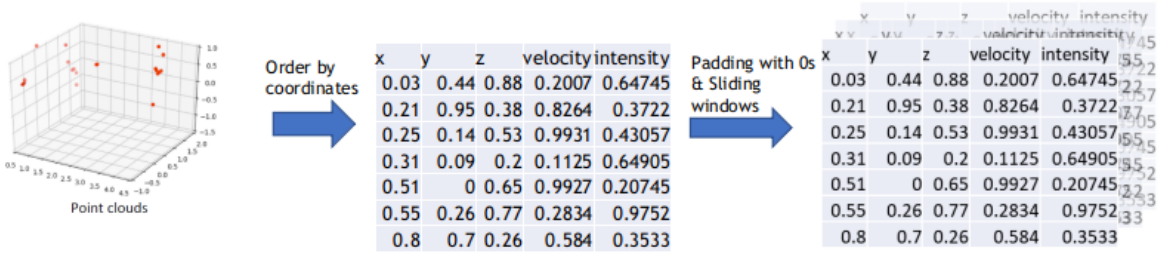


Figure 8: Preprocessing

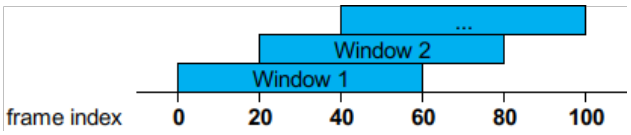


Figure 9: Sliding window

to 40 frames. The Figure 9 shows our sliding window configuration. This results in the feature of each sample having a shape of (5, 42, 60).

4.3 Classification

We evaluated the task using different classifiers:

- CNN+Transformer
- Multi-MLPs+Transformer
- Multi-CNNs+Transformer
- BiLSTM

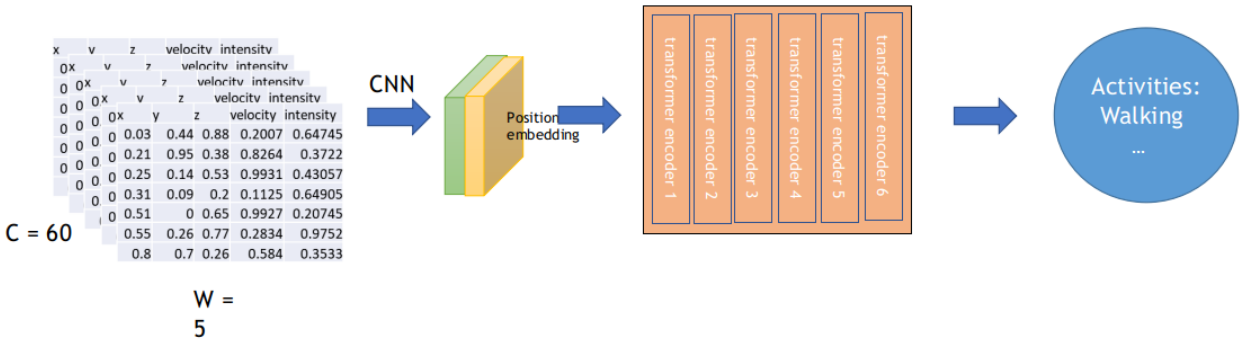


Figure 10: CNN+Transformer

The first model is CNN+Transformer (Figure 17). The first algorithm is to combine a CNN with a transformer encoder. Transformers are a type of neural network architecture gaining popularity. They were widely used in natural language processing tasks such as language translation, text to speech transformation. Nowadays, People found transformer can be applied to computer vision and usually having a very good performance. In the vision transformer architecture, the images are segmented and then patches are sent in sequence to the transformer encoder. The idea behind this model is very similar to vision transformer, except we treated the whole window as an image. For example, this window has the dimension 5 by 42 by 60. 5 is the width, 42 is the height, 60 is the channel number. CNN can be used to partition and extract features from each patches. Then all the patches along are lined with position embedding and fed into 6 transformer encoders. The classification is done at the end.

The second model is Multi-MLPs+Transformer (Figure 11). This is another hybrid architecture which combines MLP with transformer encoder. First, the data from each frame is flattened, and fed into MLP. Since the windows size is 60, 60 MLPs are used to extract features. Then the output are concatenated and send to transformer encoders.

The third model is Multi-CNNs+Transformer (Figure 12). This is similar with the previous one, except it used CNN to extract features.

The last model is BiLSTM (Figure 13). The BiLSTM model contains double LSTM chain to handle series data from early time stamp to late time stamp and from late to early. Then the output of the two chains of BiLSTM will be concatenated and do classification in the end.

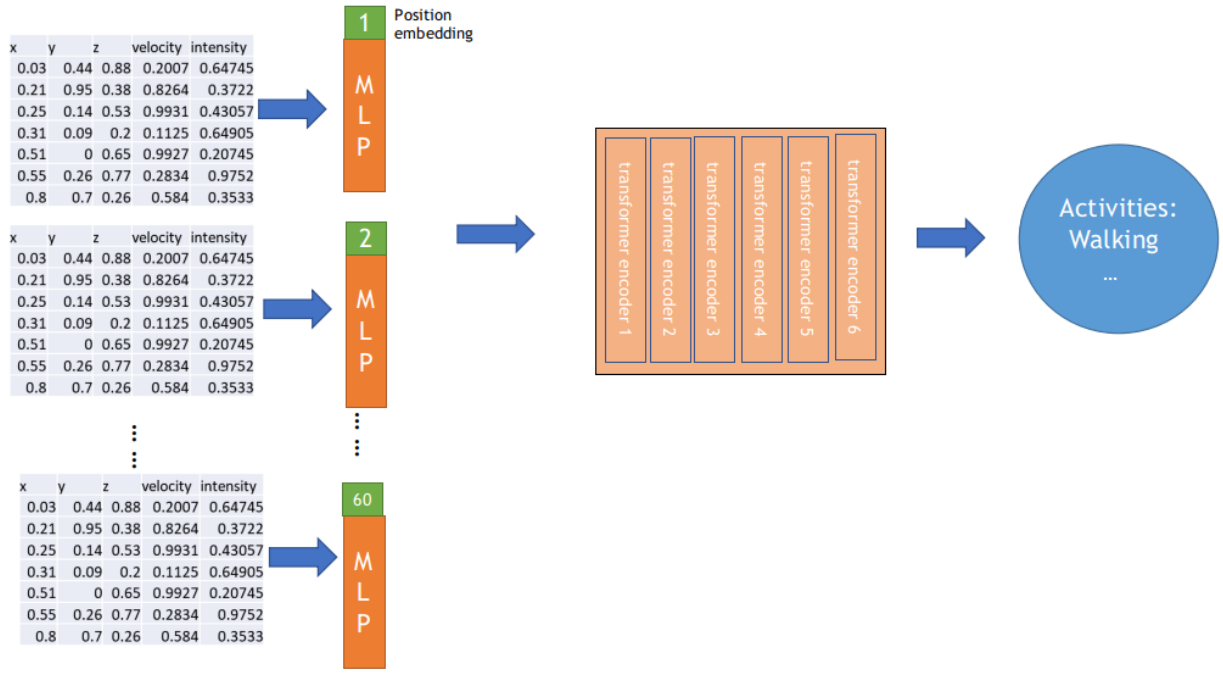


Figure 11: Multi-MLPs+Transformer

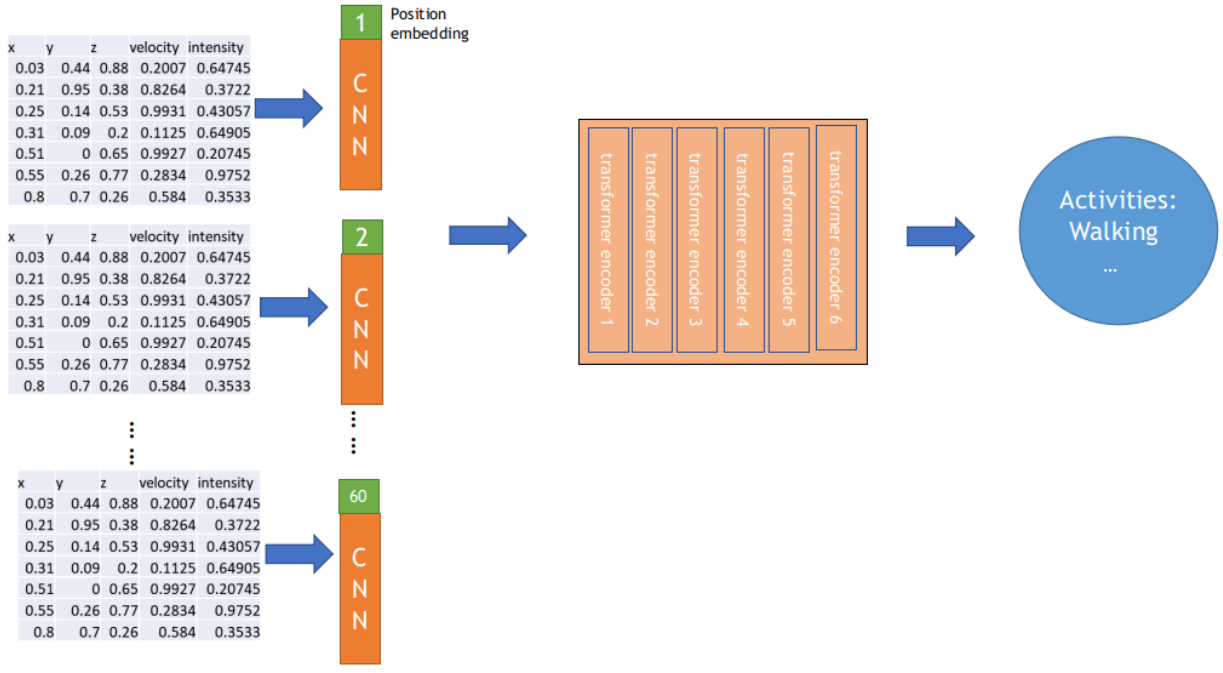


Figure 12: Multi-CNNs+Transformer

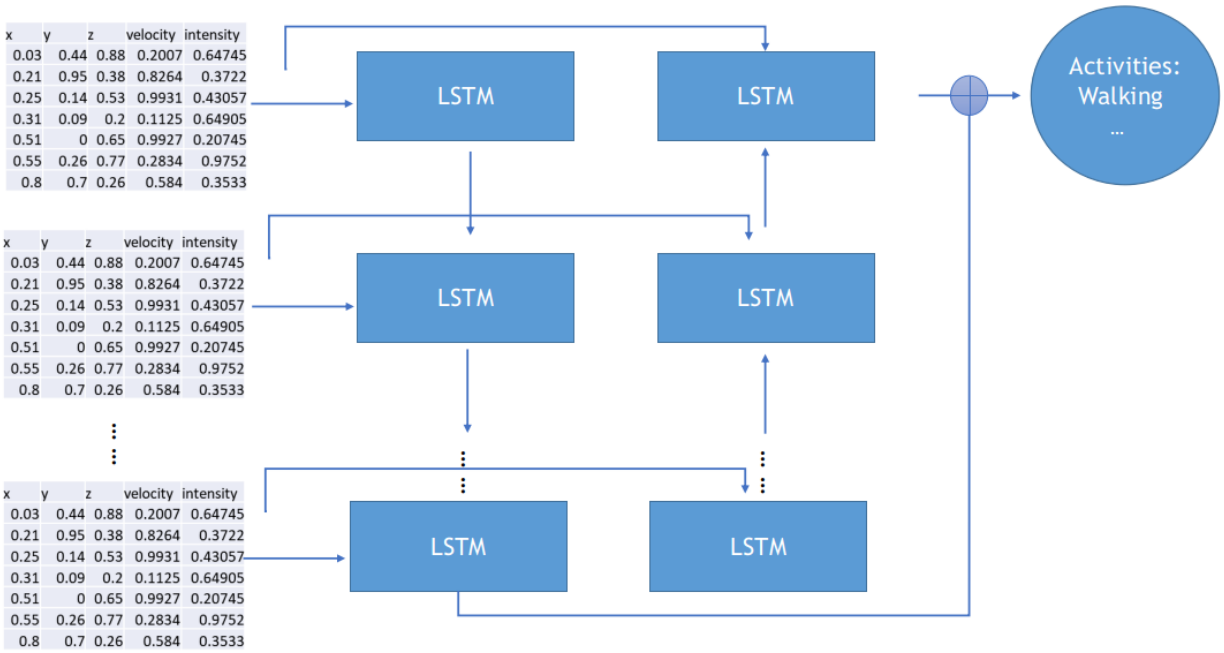


Figure 13: BiLSTM

5 Model performance

Models were trained and tested on Pronto with 64 CPUs, 64G memory and 1 GPU. First, we evaluate the overall accuracy for the four types of models proposed. The accuracy is calculated using this formula $Accuracy = 100\% \times \frac{C}{A}$ where C is the number of the samples that are correctly classified and A is the total number of samples.

Model	**Validation Accuracy**	**Test Accuracy**	**Training Speed (s/epoch)**
CNN+Transformer	95.88%	89.82%	5.97
Multi-MLPs+ Transformer	88.89%	85.67%	15.93
Multi-CNNs+ Transformer	93.00%	86.17%	12.77
BiLSTM	97.94%	91.87%	3.45

Table 2: Model performance

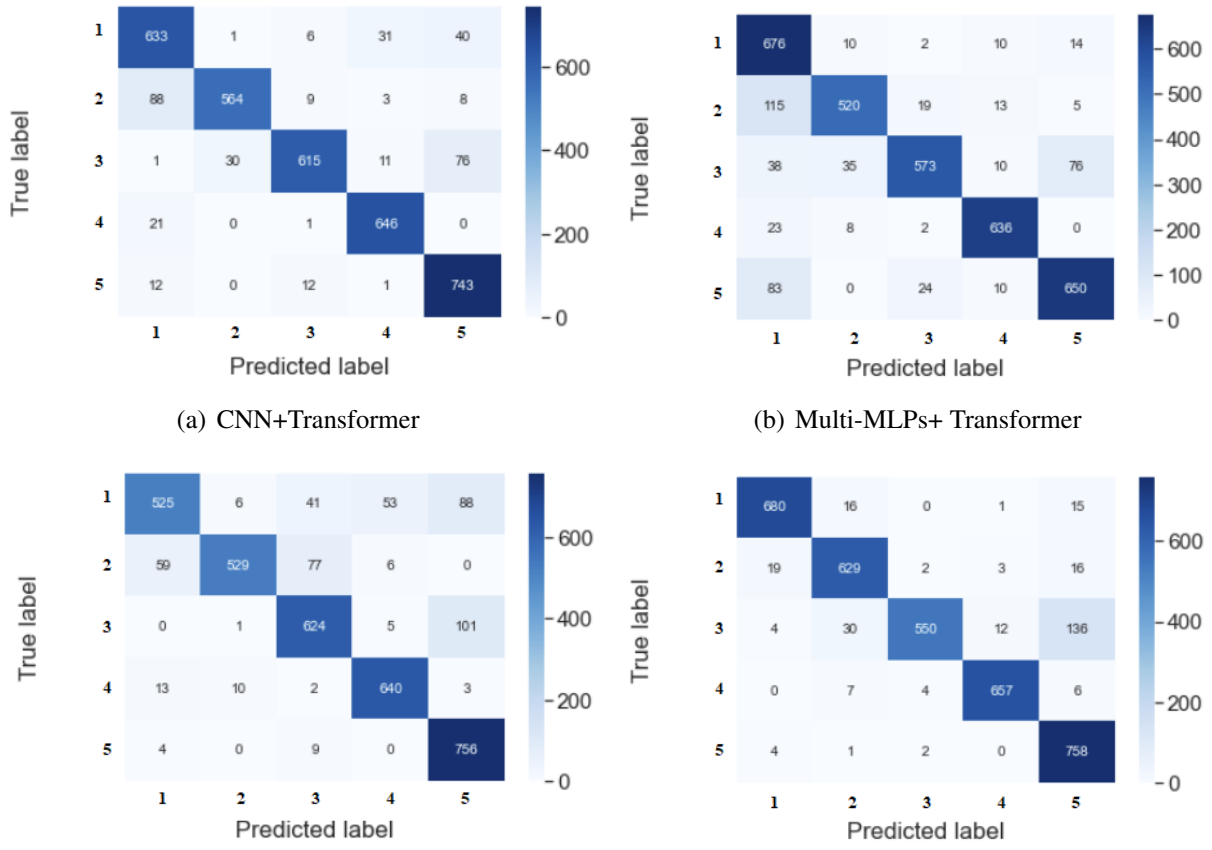
We show the accuracy over validation set, test set and the training speed at Table 2. From the result, BiLSTM model has the best accuracy on test set. It has advantage on containing more context information, since it combines the context information on both ends of the window. Following 2nd place is the CNN transformer model, which proves that the transformers based architecture can also works on our task. Given it's accuracy is very closed to BiLSTM model, we are confident that with more parameter tuning, the model can have better performance.

Label	**Activity**
1	Workout
2	Walk
3	Fall down
4	Stand up
5	Sit down

Table 3: Class label vs Activity

Next we show the confusion matrices for the four models on test set at Figure 14. The rows are the predicted label and the column are the true label of the sample. In each cell of the confusion matrix, we can see how of many samples of label i is classified as label j. See Table 3 for the class label and activity correspondence.

From the confusion matrices we can see the most common type of error across models is misclassifying fall down with sit down. This can be expected since both activities have a large movement in z-coordinate so they resemble in feature space to some extent. Another possible reason is due to the sliding window configuration, the movement in the activity is truncated which classifier hard to distinguish the two because the incomplete information. The BiLSTM model also shows a very high precision in fall down detection, only a few samples from other activity are classified as a fall down.



(a) CNN+Transformer

(b) Multi-MLPs+ Transformer

(c) Multi-CNNs+ Transformer

(d) BiLSTM

Figure 14: Confusion Matrix for four models.

6 Visualization system

While developing the classification algorithm, we find it is mandatory to devise a simplistic user interface (UI) to visualize synchronized RGB video, point cloud stream, and classification result together. The UI is used in checking out ground truth activities and predicted activities by second. Cons and pros of each deep learning model are fully exposed in check-out process. In terms of practical deployment, the current UI would be converted into a real-time version to take data streaming as input. The updated UI is allowable to deploy on AWS for data visualization.

Figure 15 shows the snipped pictures while our deep learning model is detecting targeted activities. The left window is the video generated by RGB camera, which is utilized as ground truth data for check-out. The central window displays the animation of point cloud produced by radar sensor. The animation lags behind RGB video due to the limited frequency we can set in matplotlib.animation, which will be debugged later on. The right window is the activity and its associated probability predicted by pre-trained deep learning model. For more details, please refer to Appendix C.

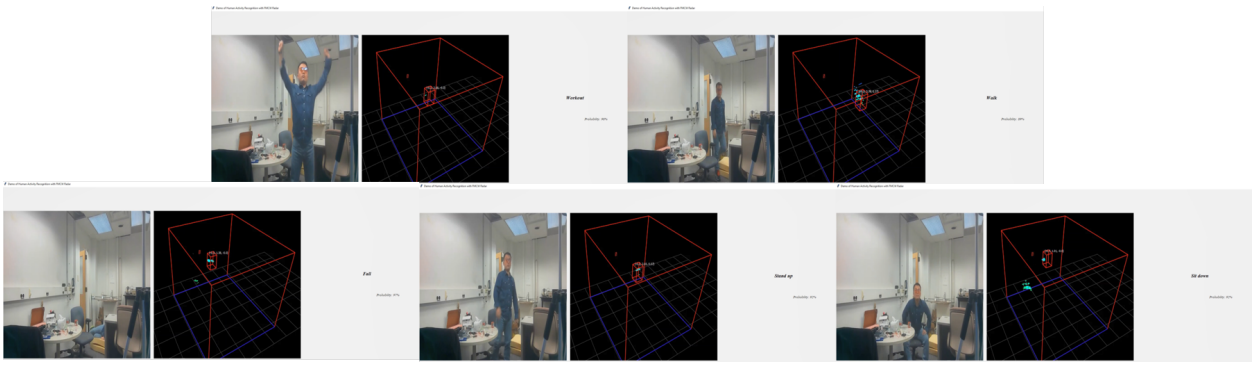


Figure 15: A series of snipped UI pictures for five activities: workout, walk, fall down, stand up, and sit down.

7 System deployment

Due to limited time, we don't have a chance to apply the system in practice. Herein, we specify a deployment plan combining Raspberry Pi and AWS IoT and elaborate the implementation details in step.

7.1 Connecting Raspberry Pi to AWS IoT

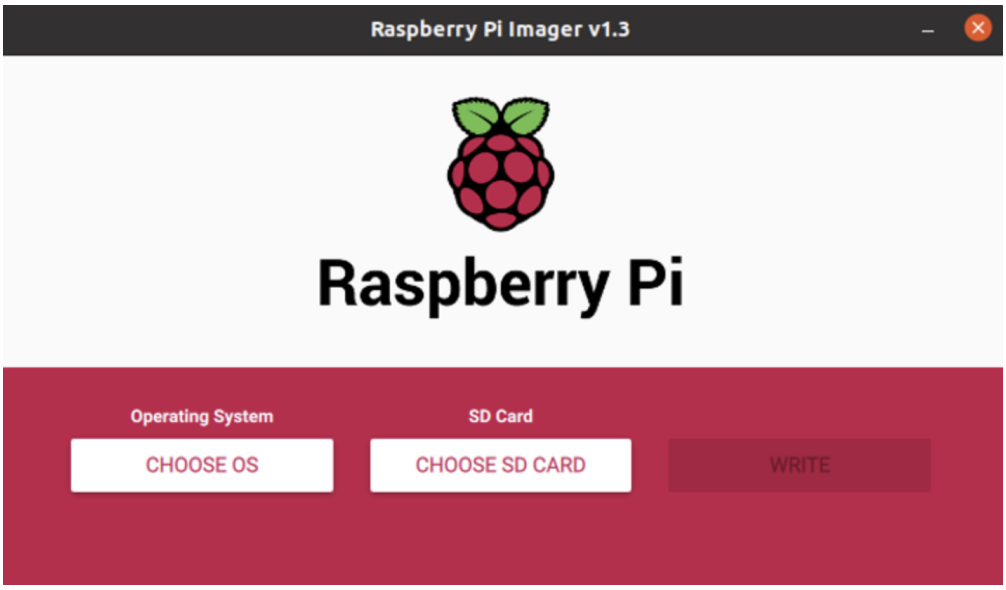


Figure 16: Raspberry Pi

- Set up Raspberry Pi and install Raspberry Pi OS
- Connect radar sensor to Raspberry Pi and configure radar sensor
- Create an IoT Thing in the AWS IoT Thing Registry
- Make a policy which gives device permissions to access AWS IoT

- Add a certificate for Raspberry Pi with public key cryptography
- Save the certificate and sync Raspberry Pi with an AWS IoT Shadow
- Reconfigure and debug Raspberry Pi if there is any bug
- Complete connection until no bug happens

7.2 Data processing, fall-down alarming, and data visualization

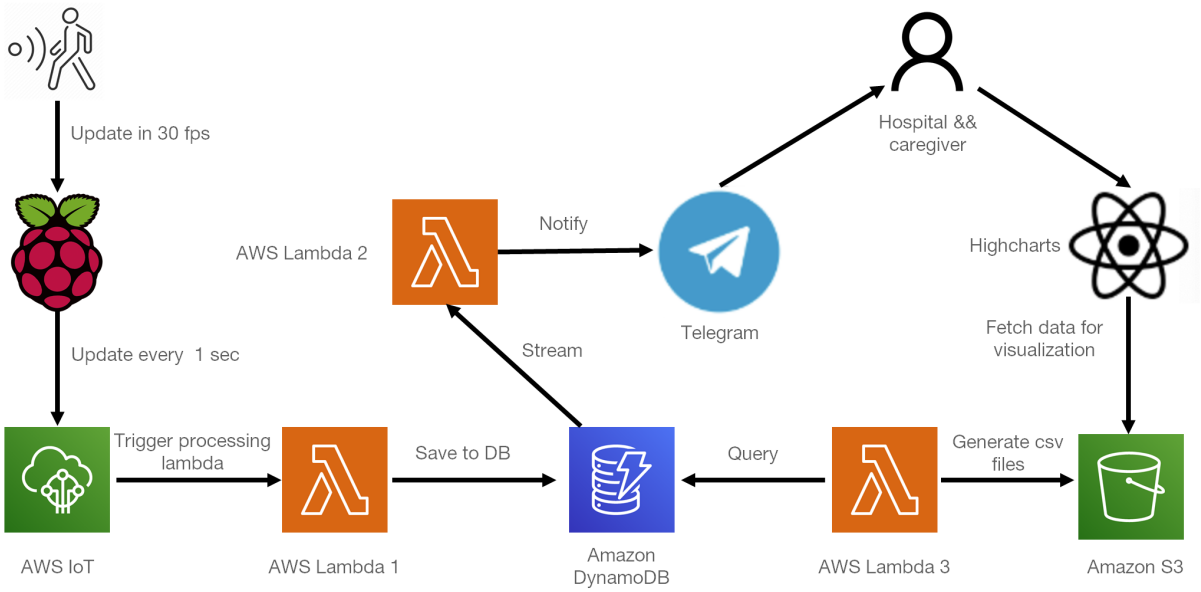


Figure 17: Roadmap of system deployment on AWS IoT

AWS IoT Core is a managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices. As long as connected to AWS IoT, Raspberry Pi is able to upload point cloud streaming in 30 times per second (corresponding to 30 fps). The uploaded data will be transferred and stored in DynamoDB permanently by ASW Lambda 1. The pre-trained deep leraning model is employed in AWS Lambda 2 for activities recognition. As long as the predicted probability of fall-down exceeds the presetting threshold, an alarming notification will be sent immediately to hospital or caregiver and call for first-aid. To visualize history data, the periodic AWS Lambda 3 is invoked to retrieve data from DynamoDB and generate CSV file into public Amazon S3 bucket. Highcharts is utilized as the visualization tool due to its friendly operation. The customized hook in Highcharts fetches data from Amazon S3 and passes it to Highcharts component for visualization.

8 Discussion

The final goal of this project is to implement a system that sending alarm whenever detecting a human fall down. Our experiments results shows the great potential of our models for indoor

activity recognition. However, the dataset only cover a small category of activities, the model's capability under larger dataset is yet to be tested. Only tested under ideal laboratory environment, there are uncertainty remained when our model is deployed in common life, factors such as people of different shape, the room's structure, multiple people in a room, etc., can effect the model's performance. Problems described above will need to tackle before the model can be put into practice.

Due to the time limit of the project, some component in the initial proposal are not fully explored or implemented yet, these will be further discussed in our future work and the appendix.

9 Conclusion and Future Work

In this project, we successfully implemented indoor human activity recognition by radar sensor. Our hybrid deep learning models including transformer architecture yielded comparable recognition accuracy to those by the state of the art models. For future work, we will collect more activity data and consider multi-sensor fusion for a more robust system. Voxelization and other preprocessing algorithms and configurations will also be experimented. Considering the popularity and the underlying potential of transformer architecture, the transformer models will be further optimized. The cloud-based service for our fall-alarm system will continue to be developed.

REFERENCES

[1] T. Buettner *et al.*, "World population prospects–a long view," *Economie et Statistique/Economics and Statistics*, no. 520-521, pp. 9–27, 2020.

[2] W. C. H. A. a Fall, "Important facts about falls," 2016.

[3] D. A. Umphred, R. T. Lazaro *et al.*, *Neurological rehabilitation*. Elsevier Health Sciences, 2012.

[4] R. Beringer, A. Sixsmith, M. Campo, J. Brown, and R. McCloskey, "The "acceptance" of ambient assisted living: Developing an alternate methodology to this limited research lens," in *International Conference on Smart Homes and Health Telematics*. Springer, 2011, pp. 161–167.

[5] W. Wang, A. X. Liu, and M. Shahzad, "Gait recognition using wifi signals," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 363–373.

[6] H. Zou, Y. Zhou, J. Yang, W. Gu, L. Xie, and C. Spanos, "Wifi-based human identification via convex tensor shapelet learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[7] A. Jalal, S. Kamal, and D. Kim, "Human depth sensors-based activity recognition using spatiotemporal features and hidden markov model for smart environments," *Journal of computer networks and communications*, vol. 2016, 2016.

[8] P. Zhao, C. X. Lu, J. Wang, C. Chen, W. Wang, N. Trigoni, and A. Markham, "mid: Tracking and identifying people with millimeter wave radar," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019, pp. 33–40.

[9] A. D. Singh, S. S. Sandha, L. Garcia, and M. Srivastava, "Radhar: Human activity recognition from point clouds generated through a millimeter-wave radar," in *Proceedings of the 3rd ACM Workshop on Millimeter-wave Networks and Sensing Systems*, 2019, pp. 51–56.

[10] Iwr1642boost evaluation board — ti.com. [Online]. Available: https://www.ti.com/tool/IWR1642BOOST

Appendix A Example Data

Figure 18 shows a snipped picture of raw data. Each consecutive 5-row data $(x, y, z, doppler, SNR)$ constitutes into a frame. (x, y, z) is the spatial position of point cloud. $doppler$ means the doppler intensity for each point cloud. $doppler = 0$ presents the received signal reflected by static environment which is supposed to be cut off during data preprocessing. SNR is termed the signal-to-noise ratio means which is related to the transmit power, propagation loss, RCS of the target and thermal noise level in the received chain and the ADC sampling rate.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	1.682294	4.821502	1.300623	0.484571	1.619761	4.631613	0.471728	1.516749	0.441001	1.292514	2.814653	0.318571	1.175241	2.526089	3.09144	3.519747	1.070118	2.819892	2.153962	2.783523	0.511597	1.260853	1.844611	0.461318	0.718043	2.367709	0.426134	2.162111	0.741453	1
2	0.83516	2.100142	0.551116	0.199637	0.950779	2.128621	0.228334	1.107735	0.283164	1.631439	4.027153	0.465653	1.717846	4.214155	5.042267	5.491294	1.785227	5.286754	4.347497	5.218569	1.280872	3.06736	4.487505	1.189384	1.907646	5.447627	1.377575	6.989518	2.679066	3
3	0.264678	0.741114	0.345679	0.221579	0.264678	1.247404	0.221579	0.264678	0.221579	-0.14594	1.202359	-0.07376	-0.14594	2.757666	-0.14594	-0.42011	1.187325	2.570632	-0.18032	0.811576	1.187325	0.553466	0.861783	2.511359	-0.18852	-0.51298	0.11125	0.476435	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	697.12	246.36	8.52	41.88	633.44	55.12	73.88	0	34.76	129.12	9.64	14.76	115.36	19.24	38.52	8	64.88	13.88	10.12	39.36	9.76	13.12	8.88	22	10	17	63.64	9.64	167.64
6	1.682294	4.842262	0.484571	1.619761	4.631613	0.471728	6.112812	1.516749	0.441001	1.292514	2.814653	0.323211	3.167625	1.175241	2.526089	3.09144	1.070118	2.819892	2.153962	2.783523	0.511597	1.260853	1.844611	0.718043	2.367709	0.426134	2.162111	0.741453	1.031356	1
7	0.83516	2.051822	0.199637	0.950779	2.128621	0.228334	2.735655	1.107735	0.283164	1.631439	4.027153	0.462444	5.052441	1.717846	4.214155	5.042267	1.785227	5.286754	4.347497	5.218569	1.280872	3.06736	4.487505	1.907646	5.447627	1.377575	6.989518	2.679066	3.219692	4
8	0.264678	0.741114	0.221579	0.264678	1.247404	0.221579	0.264678	0.221579	-0.14594	1.202359	-0.07376	-0.14594	-0.71905	-0.14594	1.202359	2.570632	-0.42011	1.187325	2.570632	-0.18032	0.811576	1.187325	0.861783	2.511359	-0.18852	-0.51298	0.11125	0.476435	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	709	255.12	44.52	715.68	60.64	78.24	8.24	0	35.12	124.76	10	15.36	8.76	109.12	20.12	40.24	64.12	16	10.52	38.76	8.24	13.12	9.12	10.36	16	48.76	9.88	168.52	27.12
11	1.682294	4.842262	0.484571	1.619761	4.631613	6.177297	0.473988	1.516749	0.443811	0.35036	1.292514	2.814653	0.323211	1.175241	2.526089	3.09144	1.070118	2.819892	2.153962	2.783523	0.511597	1.260853	1.844611	0.718043	2.367709	0.426134	2.162111	0.741453	1.031356	1
12	0.83516	2.051822	0.199637	0.950779	2.128621	2.914163	0.223605	1.107735	0.27874	0.442231	1.631439	4.027153	0.462444	1.717846	4.214155	5.042267	1.785227	5.286754	4.347497	5.218569	1.280872	3.06736	4.487505	1.907646	5.447627	1.377575	6.989518	2.679066	3.219692	4
13	0.264678	0.741114	0.221579	0.264678	1.247404	2.49321	0.221579	0.264678	0.221579	-0.07376	-0.14594	1.202359	-0.07376	-0.14594	1.202359	2.570632	-0.14594	-0.42011	1.187325	2.570632	-0.18032	0.811576	1.187325	0.861783	2.511359	-0.18852	-0.51298	0.11125	0.476435	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	732.12	319.12	41.24	1087.56	65.64	9.64	72.36	0	32.64	8.12	135.88	10.64	18.88	117	20.36	45.24	66.52	14.24	9.88	43	9	13.24	8.64	10.24	16.64	46	9.24	167.12	27
16	1.682294	4.821502	6.288892	0.484571	1.619761	4.631613	0.473988	6.112812	1.516749	0.446576	0.35036	1.292514	2.814653	0.323211	1.175241	3.131832	2.526089	3.09144	1.070118	2.872617	2.153962	2.783523	0.511597	1.260853	1.844611	0.718043	2.367709	0.426134	2.162111	0
17	0.83516	2.100142	2.664806	0.199637	0.950779	2.128621	0.223605	2.735655	1.107735	0.274288	0.442231	1.631439	4.027153	0.462444	1.717846	5.108148	4.214155	5.042267	1.785227	5.258291	4.347497	5.218569	1.280872	3.06736	4.487505	1.907646	5.447627	1.377575	6.989518	2
18	0.264678	0.741114	2.49321	0.221579	0.264678	1.247404	0.221579	2.831461	0.264678	0.221579	-0.07376	-0.14594	1.202359	-0.07376	-0.14594	-0.42011	1.202359	2.570632	-0.14594	-0.42011	1.187325	2.570632	-0.18032	0.811576	1.187325	0.861783	2.511359	-0.18852	-0.51298	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	755.76	231.64	8.24	45.12	1017.8	57.88	73.12	10	0	32.52	8.36	137.64	10.52	16.64	114.24	9.36	19.24	38.24	63.88	17.24	9.52	38.52	8.88	12.88	8.24	10.52	16.24	42.64	8.88
21	1.682294	4.842262	0.484571	1.619761	4.631613	0.473988	6.112812	1.516749	0.446576	0.35036	1.292514	2.814653	0.327819	1.175241	3.131832	2.526089	3.09144	3.519747	1.070118	2.819892	2.153962	2.783523	0.511597	1.260853	1.844611	0.718043	2.367709	0.426134	2.162111	0
22	0.83516	2.051822	0.199637	0.950779	2.128621	0.223605	2.735655	1.107735	0.274288	0.442231	1.631439	4.027153	0.459189	1.717846	5.108148	4.214155	5.042267	5.491294	1.785227	5.286754	4.347497	5.218569	1.280872	3.06736	4.487505	1.907646	5.447627	1.377575	6.989518	2
23	0.264678	0.741114	0.221579	0.264678	1.247404	0.221579	2.831461	0.264678	0.221579	-0.07376	-0.14594	1.202359	-0.07376	-0.14594	-0.42011	1.202359	2.570632	2.757666	-0.14594	-0.42011	1.187325	2.570632	-0.18032	0.811576	1.187325	0.861783	2.511359	-0.18852	-0.51298	
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	744.76	237.24	39.88	1019.56	59.64	66.64	8.88	0	33.76	8.24	127	10.36	18.12	109.64	8.76	19.76	38.76	8.64	62.76	16.88	9.52	39.24	9	13	8.24	10.52	16.52	54	9.64
26	1.682294	4.821502	0.484571	1.619761	4.631613	0.473988	1.516749	0.446576	0.35036	1.292514	0.327819	1.175241	3.131832	2.526089	3.09144	1.070118	2.819892	2.153962	2.783523	0.511597	2.57734	1.260853	1.844611	0.718043	2.367709	0.426134	2.162111	0.741453	1.031356	1
27	0.83516	2.100142	0.199637	0.950779	2.128621	0.223605	1.107735	0.274288	0.442231	1.631439	0.459189	1.717846	5.108148	4.214155	5.042267	1.785227	5.286754	4.347497	5.218569	1.280872	6.847291	3.06736	4.487505	1.907646	5.447627	1.377575	6.989518	2.679066	3.219692	4
28	0.264678	0.741114	0.221579	0.264678	1.247404	0.221579	0.264678	0.221579	-0.07376	-0.14594	-0.07376	-0.14594	-0.42011	1.202359	2.570632	-0.14594	-0.42011	1.187325	2.570632	-0.18032	0.811576	1.187325	0.861783	2.511359	-0.18852	-0.51298	0.11125	0.476435	1
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	741.64	252.36	43.76	1025.32	55.52	71.12	0	30.24	9.12	129.64	19	108	10.52	19.88	44.88	61.52	19.52	10.12	43.88	9.88	8.12	13.12	8.64	10.24	18	48.24	10.52	167.88	26.24
31	1.682294	4.842262	0.484571	1.619761	4.631613	0.473988	6.112812	1.516749	0.446576	0.35036	1.292514	0.323211	1.175241	3.131832	2.526089	3.09144	1.070118	2.872617	2.153962	2.783523	0.511597	1.260853	1.844611	0.718043	2.367709	0.426134	2.162111	0.741453	1.031356	1
32	0.83516	2.051822	0.199637	0.950779	2.128621	0.223605	2.735655	1.107735	0.274288	0.442231	1.631439	0.462444	1.717846	5.108148	4.214155	5.042267	1.785227	5.258291	4.347497	5.218569	1.280872	3.06736	4.487505	1.907646	5.447627	1.377575	6.989518	2.679066	3.219692	4
33	0.264678	0.741114	0.221579	0.264678	1.247404	0.221579	2.831461	0.264678	0.221579	-0.07376	-0.14594	-0.42011	1.202359	2.570632	-0.14594	-0.42011	1.187325	2.570632	-0.18032	0.811576	1.187325	0.861783	2.511359	-0.18852	-0.51298	0.11125	0.476435	1
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	762.36	266.12	51.12	1383.92	57	95.88	8.36	0	37.88	8.36	129.36	20.12	113.24	10.52	18.88	37.76	65.64	19	9.64	36.24	9.76	13.24	8.52	10	15.88	48.36	10	167.52	26.88
36	1.682294	4.842262	0.484571	1.619761	4.631613	0.473988	6.112812	1.516749	0.443811	0.35036	1.292514	2.814653	0.323211	1.175241	3.131832	2.526089	3.09144	1.070118	2.872617	2.153962	2.783523	0.511597	2.57734	1.260853	1.844611	0.718043	2.367709	0.426134	2.162111	0
37	0.83516	2.051822	0.199637	0.950779	2.128621	0.223605	2.735655	1.107735	0.27874	0.442231	1.631439	4.027153	0.462444	1.717846	5.108148	4.214155	5.042267	1.785227	5.258291	4.347497	5.218569	1.280872	6.847291	3.06736	4.487505	1.189384	1.907646	5.447627	6.989518	2
38	0.264678	0.741114	0.221579	0.264678	1.247404	0.221579	2.831461	0.264678	0.221579	-0.07376	-0.14594	1.202359	-0.07376	-0.14594	-0.42011	1.202359	2.570632	-0.14594	-0.42011	1.187325	2.570632	-0.18032	-0.51298	0.811576	1.187325	0.554466	0.861783	2.511359	-0.51298
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	772.76	282	47.76	1151.92	59.36	87.12	8.12	0	35.12	8.24	134.52	10.52	18.24	114.76	10.36	19.64	40.24	64.64	19.76	9.88	38.64	10.52	9.12	13.36	8.64	19.12	10.12	17.36	11.76
41	1.682294	4.821502	0.484571	1.619761	4.631613	0.471728	6.112812	1.516749	0.441001	1.292514	2.814653	0.318571	1.175241	3.131832	2.526089	3.09144	3.519747	1.070118	2.872617	2.153962	2.783523	0.511597	1.260853	1.844611	0.718043	2.367709	0.420756	2.162111	0.741453	1
42	0.83516	2.100142	0.199637	0.950779	2.128621	0.228334	2.735655	1.107735	0.283164	1.631439	4.027153	0.465653	1.717846	5.108148	4.214155	5.042267	5.491294	1.785227	5.258291	4.347497	5.218569	1.280872	3.06736	4.487505	1.907646	5.447627	1.333518	6.989518	2.679066	3
43	0.264678	0.741114	0.221579	0.264678	1.247404	0.221579	2.831461	0.264678	0.221579	-0.14594	1.202359	-0.07376	-0.14594	2.757666	-0.14594	-0.42011	1.202359	2.570632	-0.18032	0.811576	1.187325	0.861783	2.511359	-0.18032	-0.51298	0.11125	0

pcBufPing

Figure 18: A snippet of raw data

Appendix B Code Snippet

Herein, we show some code snippet for data preprocessing, deep learning model building, point cloud animation and user interface, which is detailed in Listing 1, Listing 2,Listing 3, and Listing 4 respectively.

Listing 1: Example source code for data preprocessing.

```
import glob
import os
import numpy as np


def get_data(file_path):
   with open(file_path) as f:
      lines = f.readlines()

   frame_num_count = -1
   frame_num = []
   x = []
   y = []
   z = []
   velocity = []
   intensity = []
   wordlist = []


   for x1 in lines:
      for word in x1.split():
         wordlist.append(word)

   length1 = len(wordlist)

   for i in range(0,length1):
      if wordlist[i] == "point_id:" and wordlist[i+1] == "0":
         frame_num_count += 1
      if wordlist[i] == "point_id:":
         frame_num.append(frame_num_count)
      if wordlist[i] == "x:":
         x.append(wordlist[i+1])
      if wordlist[i] == "y:":
         y.append(wordlist[i+1])
      if wordlist[i] == "z:":
         z.append(wordlist[i+1])
      if wordlist[i] == "velocity:":
         velocity.append(wordlist[i+1])
      if wordlist[i] == "intensity:":
         intensity.append(wordlist[i+1])
```

Listing 2: Example source code for model building.

```
def full_3D_model(input_x, input_y, reg = 0, num_feat_map = 16, summary=False)
    :
    print('building␣the␣model␣...␣')
    model = Sequential()
    # 1st layer group
    model.add(TimeDistributed(Conv3D(32, (3, 3, 3), strides=(1, 1, 1), name="
        conv1a", input_shape=(10, 32, 32,1), padding="same", activation="relu")
        ))
    # 2nd layer group
    model.add(TimeDistributed(Conv3D(32, (3, 3, 3), strides=(1, 1, 1), name="
        conv1b", padding="same", activation="relu")))

    model.add(TimeDistributed(MaxPooling3D(name="pool1", strides=(2, 2, 2),
        pool_size=(2, 2, 2), padding="valid")))

    # 3rd layer group
    model.add(TimeDistributed(Conv3D(32, (3, 3, 3), strides=(1, 1, 1), name="
        conv2a", padding="same", activation="relu")))
    model.add(TimeDistributed(Conv3D(32, (3, 3, 3), strides=(1, 1, 1), name="
        conv2b", padding="same", activation="relu")))
    model.add(TimeDistributed(MaxPooling3D(strides=(2, 2, 2), pool_size=(2, 2,
        2), data_format="channels_first", name="pool2", padding="valid")))

    model.add(TimeDistributed(Conv3D(32, (3, 3, 3), strides=(1, 1, 1), name="
        conv2a", padding="same", activation="relu")))
    model.add(TimeDistributed(Conv3D(32, (3, 3, 3), strides=(1, 1, 1), name="
        conv2b", padding="same", activation="relu")))
    model.add(TimeDistributed(MaxPooling3D(strides=(2, 2, 2), pool_size=(2, 2,
        2), data_format="channels_first", name="pool2", padding="valid")))


    model.add(TimeDistributed(Flatten()))
    model.add(Dropout(0.5))

    model.add(Bidirectional(LSTM(16, return_sequences=False, stateful=False)))

    model.add(Dropout(.3))

    model.add(Dense(input_y.shape[1], activation='softmax', name = 'output'))

    return model
```

Listing 3: Example source code for point cloud animation.

```
#To plot points cloud on our data
import ast
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

file_name = 'data_IWR6843_modified.cvs'

s = open(file_name, 'r').read()
data = ast.literal_eval(s) # It is a 3D list.

#Get part of the data
def get_data(data,start_i, end_i):
   data2 = []
   for i in range(start_i,end_i):
      data2.append(data[i])
   return data2

def animate(i):
   datai = data[i]
   xdata = datai[0]
   ydata = datai[1]
   zdata = datai[2]
   ax.clear()

   ax.scatter3D(xdata, ydata, zdata, c="blue")
   title = 'The frame No: {}'.format(i)
   ax.set_title(title)
   ax.set_xlabel('X')
   ax.set_ylabel('Y')
   ax.set_zlabel('Z')

   ax.set_xlim(ranges[0])
   ax.set_ylim(ranges[1])
   ax.set_zlim(ranges[2])
```

Listing 4: Example source code for user interface.

```
class App:
   def __init__(self, window, video_source1, video_source2):
      self.window = window
      self.window.title("Demo␣of␣human␣activity␣recognition␣with␣FMCW␣Radar")
      self.video_source1 = video_source1
      self.video_source2 = video_source2
      self.photo1 = ""
      self.photo2 = ""

      # open video source
      self.vid1 = MyVideoCapture(self.video_source1, self.video_source2)

      # Create a canvas that can fit the above video source size
      self.canvas1 = tkinter.Canvas(window, width=700, height=700)
      self.canvas2 = tkinter.Canvas(window, width=700, height=700)
      self.canvas3 = tkinter.Canvas(window, width=700, height=700)
      self.canvas1.pack(padx=5, pady=10, side="left")
      self.canvas2.pack(padx=5, pady=60, side="left")
      self.canvas3.pack(padx=5, pady=110, side="left")
      # After it is called once, the update method will be automatically
         called every delay milliseconds
      self.delay = 3
      self.count = 0
      self.update()

      self.window.mainloop()

   def update(self):
      # Get a frame from the video source
      ret1, frame1, ret2, frame2 = self.vid1.get_frame
      if ret1 and ret2:

            self.photo1 = PIL.ImageTk.PhotoImage(image=PIL.Image.fromarray(
               frame1))
            self.photo2 = PIL.ImageTk.PhotoImage(image=PIL.Image.fromarray(
               frame2))
            self.canvas1.create_image(0, 0, image=self.photo1, anchor=tkinter.
               NW)
            self.canvas2.create_image(0, 0, image=self.photo2, anchor=tkinter.
               NW)
            self.canvas3.delete('all')
            self.canvas3.create_text(300,300,font="Times␣17␣italic␣bold",text=
               activities[self.count])
            self.canvas3.create_text(400,400,font="Times␣11␣italic",text='
               Probability:␣' + percentages[self.count])
            self.count +=1

      self.window.after(self.delay, self.update)
```

Appendix C User Interface

To validate the functionality of user interface (UI) design, we take a snipped video of UI while performing the five movement: workout, walk, fall down, stand up, and sit down. The left window is the real-time video generated by RGB camera, which is utilized as ground truth data for check-out. The central window displays the animation of point cloud produced by radar sensor. The animation lags behind RGB video due to the limited frequency we can set in matplotlib.animation, which will be debugged later on. The right window is the activity and its associated probability predicted by pre-trained deep learning model.

Figure 19, Figure 20, Figure 21, Figure 22, and Figure 23 are the snipped pictures for workout, walk, fall down, stand up, and sit down respectively.

If you want to watch the full video demo, please click this link: User interface video demo.

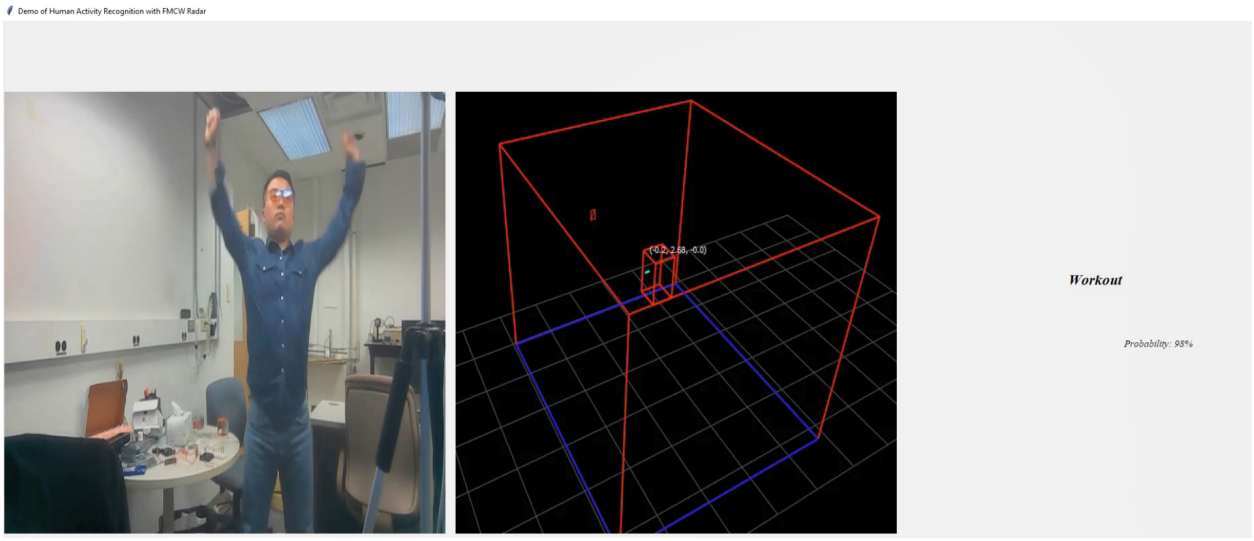


Figure 19: Workout

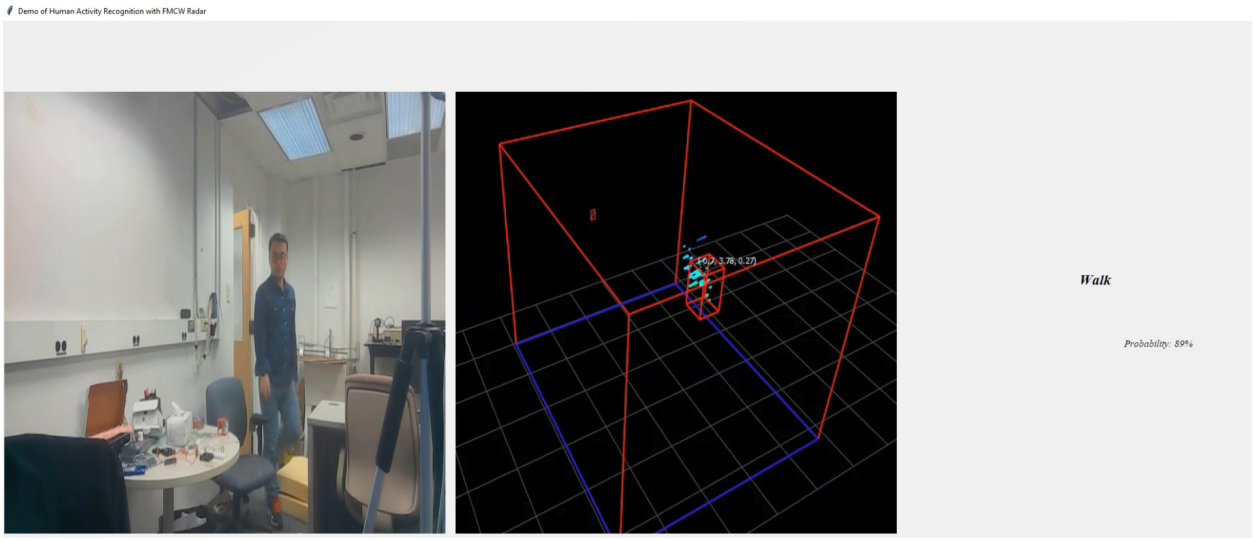


Figure 20: Walk

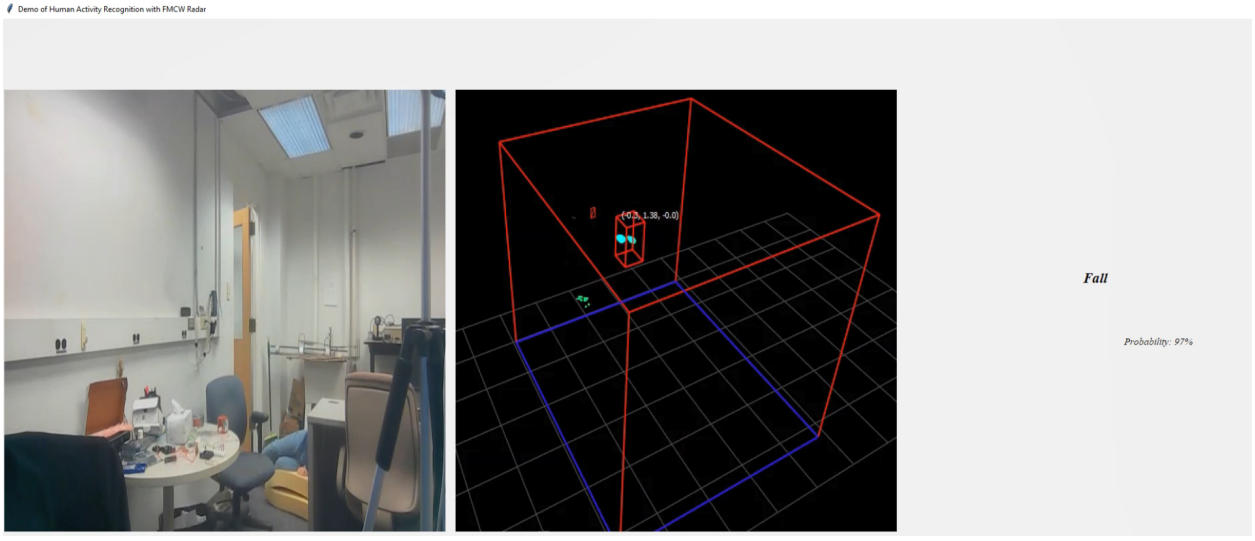


Figure 21: Fall down

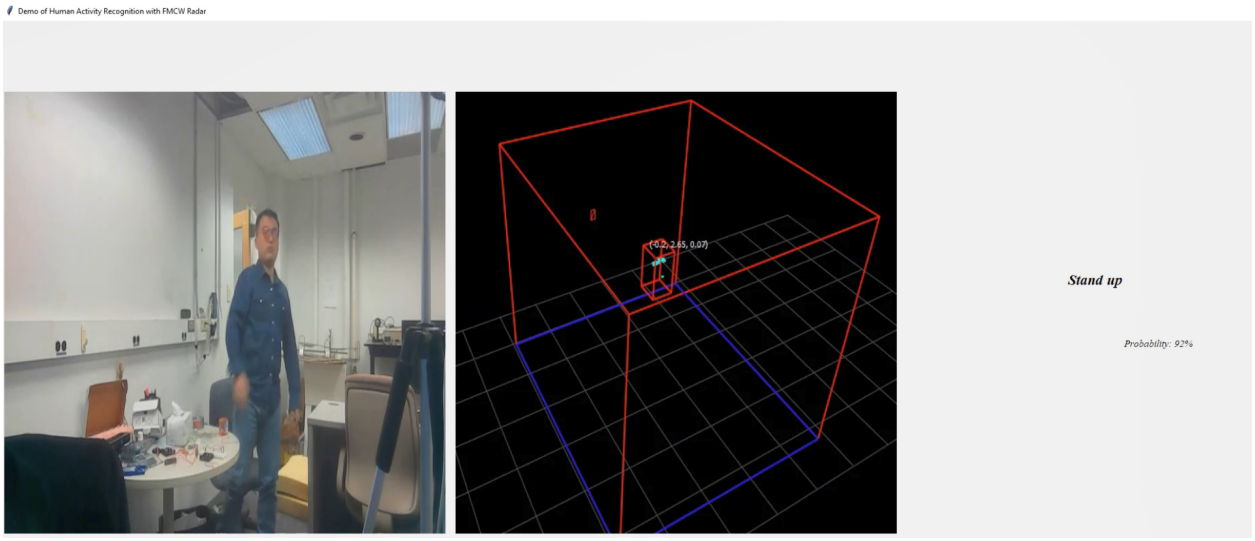


Figure 22: Stand up

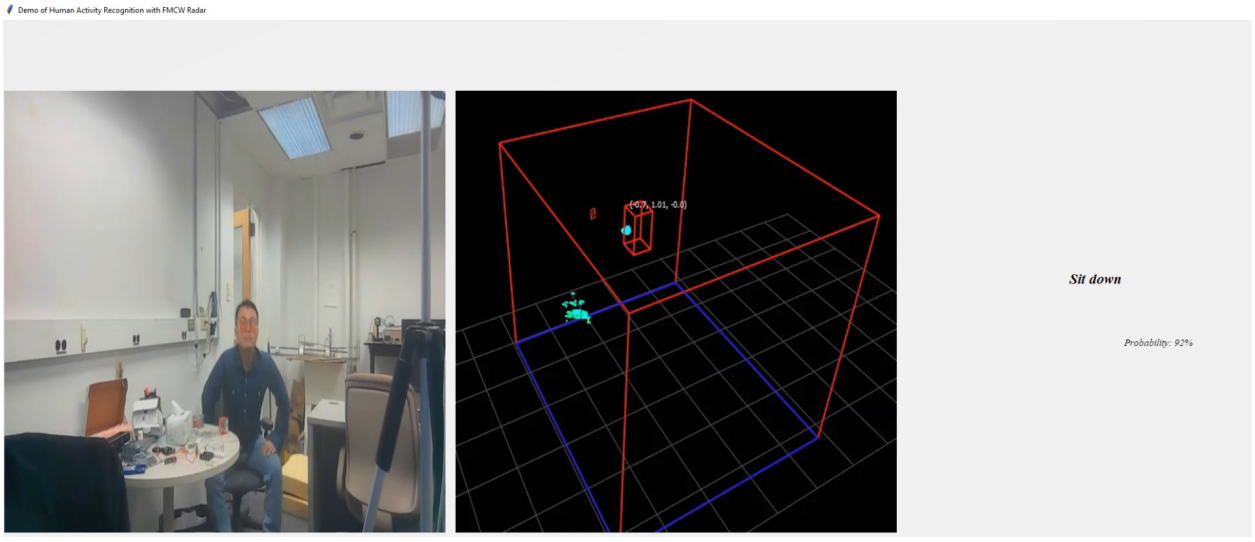


Figure 23: Sit down