CubeLearn: End-to-End Learning for Human Motion Recognition From Raw mmWave Radar Signals

Peijun Zhao, Chris Xiaoxuan Lu, Bing Wang, Niki Trigoni, and Andrew Markham

***Abstract*—mmWave FMCW radar has attracted a huge amount of research interest for human-centered applications in recent years, such as human gesture and activity recognition. Most existing pipelines are built upon conventional discrete Fourier transform (DFT) preprocessing and deep neural network classifier hybrid methods, with a majority of previous works focusing on designing the downstream classifier to improve overall accuracy. In this work, we take a step back and look at the preprocessing module. To avoid the drawbacks of conventional DFT preprocessing, we propose a complex-weighted learnable preprocessing module, named CubeLearn, to directly extract features from raw radar signal and build an end-to-end deep neural network for mmWave FMCW radar motion recognition applications. Extensive experiments show that our CubeLearn module consistently improves the classification accuracies of different pipelines, especially, benefiting those simpler models, which are more likely to be used on edge devices due to their computational efficiency. We provide ablation studies on initialization methods and structure of the proposed module, as well as an evaluation of the running time on PC and edge devices. This work also serves as a comparison of different approaches toward data cube slicing. Through our task-agnostic design, we propose a first step toward a generic end-to-end solution for radar recognition problems.**

***Index Terms*—End-to-end neural network, mmWave radar, motion recognition.**

I. Introduction

MMWAVE FMCW radar was mainly used on high-end cars and military vehicles many years ago, due to its bulky size and high cost. With the recent development in low-cost single-chip mmWave radar (e.g., TI mmWave sensors), more and more possibilities have been explored in indoor applications, such as vital sign monitoring [1], gesture recognition [2], [3], fall detection [4], [5], and human activity recognition (HAR) [6], [7], [8]. As mmWave radars are less obtrusive than cameras, they have significant potential for domestic applications, where concerns about privacy dominate.

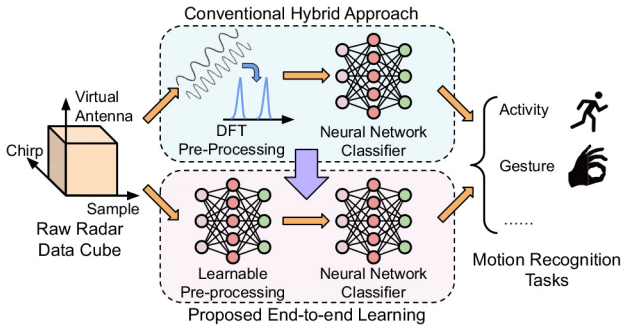


Fig. 1. We propose CubeLearn, a learnable preprocessing module to replace conventional DFT and build end-to-end deep neural networks for mmWave FMCW radar motion recognition tasks.

Typical methods for FMCW radar gesture or activity recognition follow a two-stage pipeline: radar signal preprocessing, typically with discrete Fourier transform (DFT), and a data-driven classifier for task-specific recognition purposes. In the classical radar processing chain, there are a number of different levels of data representations, ranging from Range Profile signatures, Range–Doppler, and Range–Angle maps,[1] micro-Doppler signature, to point clouds. For gesture recognition and activity recognition tasks, maps and point clouds are the most widely adopted data representations [9], [10]. Point cloud generation relies on handcrafted parameters, and generating point clouds requires receiver array which is not available in some radar configurations (e.g., SISO radar). Besides, point cloud can hardly capture fine-grained movements like finger gestures, which makes it a less universal approach. As a result, most previous works are based on maps + neural network classifier hybrid pipelines.

DFT serves as a very efficient way to extract Range, Doppler, and Angle-of-Arrival (AoA) information from the raw radar signal to generate different maps. However, it also

Manuscript received 31 July 2022; revised 16 December 2022; accepted 9 January 2023. Date of publication 17 January 2023; date of current version 7 June 2023. *(Corresponding author: Chris Xiaoxuan Lu.)*

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by The Computer Science Departmental Research Ethics Committee (CS-DREC), University of Oxford under Application No. CS_C1A_021_018.

Peijun Zhao was with the Department of Computer Science, University of Oxford, OX1 3BW Oxford, U.K. He is now with the Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: peijun.zhao@cs.ox.ac.uk; zhaoymn@mit.edu).

Chris Xiaoxuan Lu is with the Department of Informatics, University of Edinburgh, EH8 9AB Edinburgh, U.K. (e-mail: xiaoxuan.lu@ed.ac.uk).

Bing Wang was with the Department of Computer Science, University of Oxford, OX1 3BW Oxford, U.K. He is now with the Department of Aeronautical and Aviation Engineering, Hong Kong Polytechnic University, Hong Kong (e-mail: bingwang@polyu.edu.hk).

Niki Trigoni and Andrew Markham are with the Department of Computer Science, University of Oxford, OX1 3BW Oxford, U.K. (e-mail: niki.trigoni@cs.ox.ac.uk; andrew.markham@cs.ox.ac.uk).

Digital Object Identifier 10.1109/JIOT.2023.3237494

[1]We use the term "maps" to refer to 2-D and 3-D representations, which will be introduced in Section IV in detail.

has limitations like using nonadaptive basis and resolution limitations. To overcome such limitations, researchers have been trying to use end-to-end deep neural networks for radar applications in recent years. The difficulty, here, lies in that the raw radar ADC samples are complex values corresponding to the downmixed baseband, and both the magnitude and the phase are important. In previous works on end-to-end learning for radar applications, either only the real part is used [11], [12], which loses some information; or the real part and the imaginary part are treated as two channels [13], which loses the physical meaning, and using a specific network structure could make it less universal.

In this work, we explore replacing DFT preprocessing with a complex-weighted learnable preprocessing module named CubeLearn to improve motion classification accuracy, which is evaluated on human gesture and activity recognition tasks. To the authors' best knowledge, this is the first work to use stacked complex linear layers to replace DFT preprocessing for FMCW radar pipelines, and the first in-depth evaluation of a purely end-to-end radar network. Our intuition is that by making the entire network learnable and data driven, we can allow the network to better focus on important radar features, leading to higher overall accuracy. In particular, significant performance improvements are observed for those simpler models, indicating that our proposed CubeLearn module could be, especially, helpful for low-end radars (e.g., with single Tx/Rx or limited memory/computational ability) to achieve recognition performance close to a more sophisticated radar. Besides, compared to previous end-to-end works where the proposed methods are designed for specific applications, our work proposes a more generic method for different radar recognition tasks.

Our contributions in this work are listed as follows.

1) We propose CubeLearn, a learnable preprocessing module based on complex neural networks to replace the conventional DFT preprocessing, allowing the network to focus on task-specific radar signatures.
2) Through extensive evaluation, we show that our proposed method can increase the accuracy of different gesture/activity recognition pipelines and accelerate the training process.
3) We also provide a detailed, in-depth exploration of the impact of different preprocessing and classifier combinations, in terms of task accuracy and computational load on both PC and Raspberry Pi devices.
4) We release the code of the proposed CubeLearn module and our own collected data set to encourage further research on this topic.[2]

In the remainder of this article, we first discuss related work, mmWave radar background, and commonly seen pipelines. Then, we present our proposed CubeLearn module in Section V. In Section VI, we introduce the data set collection and experiment configuration, followed by evaluation in Section VII. In the remaining sections, we provide extensive ablation study and complexity analysis, and a discussion on model selection, robustness, and possible ways to utilize the elevation data as well. Section XII finally concludes this article.

[2]https://github.com/zhaoymn/cubelearn

TABLE I
MAP-BASED MMWAVE FMCW RADAR GESTURE AND ACTIVITY RECOGNITION. R: RANGE; A: ANGLE; D: DOPPLER; AND T: TIME

Pre-processing		Classifier	Related Works
1D Profile	R + A + D	LSTM	[14]
2D Map	D-T	2DCNN	[7], [15]–[18]
	R-D	2DCNN-LSTM	[3], [19]
	R-D	2DCNN-TCN	[20]
	R-D	ANN	[21]
	R-D + R-A	2DCNN-LSTM	[22]
	R-T + D-T	2DCNN	[23]
	R-T + A-T	2DCNN	[24]
3D Map	R-D-T	3DCNN	[25], [26]
	R-D-A	2DCNN + LSTM	[27]

II. RELATED WORK

A. mmWave Radar Map-Based Human Gesture/Activity Recognition

mmWave radar-based human gesture/activity recognition is one of the most studied areas in mmWave sensing. Main-stream recognition methods are based on 1-D/2-D/3-D data representations. Recent works on mmWave radar-based human gesture/activity recognition are summarized in Table I. As introduced in Section III, conventional DFT preprocessing is used to extract Range (R), Doppler (D), and AoA (A) information from the raw radar data cube. In most cases, the Range–Doppler Map, Range–Angle Map, or micro-Doppler signature is directly used as neural network input [7], [15], [16], [17], [18], [19], [20], [22], [23], [24], [25], [26], as they are fixed sized and can be processed similarly to images. Other works manually extract features and feed them into the downstream classifier [14], [21], [27].

There are many types of 2-D and 3-D data representations, such as the Range–Doppler Map used in [3], [19], [20], [21], and [22]. To form a Range–Doppler map, we need to transmit multiple chirps, and the duration is called coherent processing interval (CPI). The CPI dimension (T) can be used as a separate dimension for the neural network to extract temporal information [3], [19], [20], [22], [27], or used together with other dimensions to be processed as part of the data map [15], [16], [25], [26]. In our work, we consider different map-based pipelines in previous works, together with other possible combinations, as our baselines.

B. End-to-End mmWave Radar Gesture and Activity Recognition Methods

Researchers have also been trying to build end-to-end neural networks for radar applications. The main difficulty of designing an end-to-end structure is to handle the complex input. The magnitude and phase information are both important in the raw IF signal, as we need to extract the distance, velocity, and AoA information from the frequency and phase of the raw signal.

Sakamoto et al. [28] proposed converting the I/Q data received by CW radar receiver into an image for hand gesture recognition (HGR) with convolutional neural networks.

Complex-valued convolutional neural networks are also proposed to extract information from radar micro-Doppler signatures for activity recognition [29]. Zhao et al. [13] proposed treating the real and imaginary part as two separate channels which are processed similarly to images with real-valued convolutional layers for human activity classification. There are also some works that use real-valued input. Stadelmayer and Santra proposed using a parametric convolutional neural network (2-D Sinc Filter and 2-D Wavelet Filter) [11] to extract Range and Doppler information from raw radar data. Ye et al. proposed using two real-valued convolutional layers with Fourier initialization for human-activity classification [12], [30] on continuous-wave (CW) radar data. However, their methods lose half of the information from the imaginary part.

To the author's best knowledge, this work is the first to use stacked complex linear layers to replace the conventional DFT preprocessing to directly extract information from raw mmWave FMCW radar data, and to build end-to-end deep neural network together with downstream deep classifier which directly takes raw radar complex data cube as input for recognition tasks.

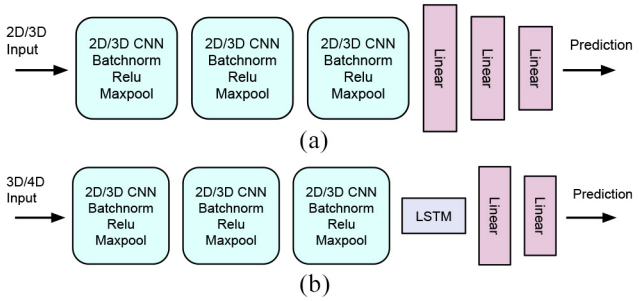


Fig. 2. Two types of downstream classifier architectures we use in this work. (a) CNN classifier. (b) CNN-LSTM classifier.

TABLE II
MAP-BASED MMWAVE FMCW RADAR GESTURE AND ACTIVITY RECOGNITION

Data Cube Slicing	Pre-processing	Classifier
first Tx-Rx pair and first chirp	R-T	2DCNN
first Tx-Rx pair, range aggregation	D-T	2DCNN
first chirp, range aggregation	A-T	2DCNN
first Tx-Rx pair	R-D-T	3DCNN/2DCNN-LSTM
first chirp	R-A-T	3DCNN/2DCNN-LSTM
whole cube, range aggregation	D-A-T	3DCNN/2DCNN-LSTM
whole cube	R-D-A-T	3DCNN-LSTM

III. MMWAVE FMCW RADAR BACKGROUND

In a typical FMCW radar configuration, a CPI consists of multiple chirps, which are short periods of signals whose frequency increases linearly with time. We use Texas Instruments IWR6843 mmWave radar in this work, which features a three transmitter, four receiver MIMO antenna array, but we note that our architecture is easily generalizable to different antenna configurations. For each chirp, typically, a single transmitter is activated (with the TDM-MIMO [31] principle), and the received signals at each receiver antenna are mixed with the transmitting signal followed by low-pass filters to produce an intermediate frequency (IF) signal that is further sampled by analog–digital converter (ADC). For each CPI, mmWave radar generates a raw data cube, with the three axes, namely, corresponding to: samples in a chirp, chirp loops, and Tx–Rx pairs.

In conventional processing, DFT is used to extract Range, Doppler, and AoA information along the three dimensions of the raw radar data cube, respectively. As the transmitting frequency increases linearly with time during a chirp, the reflection from a target at a certain distance introduces a corresponding frequency component in the IF signal, which equals to the round trip time multiplied by the frequency slope of the chirp. We can estimate the distance to different targets by extracting different frequency components with DFT, which produces a "range profile." The phase of a peak corresponding to a certain target in the Range profile changes across chirps if the target is moving, due to the slight round-trip distance change, and we can apply another DFT across chirps to extract the phase variation in order to infer the radial velocity of the object, which is often called "Doppler DFT." Furthermore, the AoA of the target introduces phase differences at the antennas in the receiver array because of the slight difference in the total length the signal travels. This can also be estimated with DFT, which is called "angle DFT." Transforming the information from the sampled time/space domain to the frequency domain is the key for extracting Range, Doppler, and AoA information in FMCW radar data processing.

IV. GESTURE AND ACTIVITY RECOGNITION MODELS

As discussed in the previous section, the mmWave radar is able to produce a raw data cube for each CPI, with three axes as sample, chirp, and virtual antenna, respectively, and Range, Doppler, and AoA information can be extracted from these three dimensions. To represent the information on these three dimensions. We often adopt maps as a straightforward data representation. Peaks in the maps correspond to detected targets. For example, a peak in the 2-D Range–Doppler maps represents a target at a certain distance with a certain radial velocity. Since in this work, we are studying gesture/activity recognition, which typically lasts several CPIs, we have an additional CPI dimension. As a result, with different combinations of information from three dimensions in the data cube, together with the CPI dimension, we supply 2-D, 3-D, or 4-D data as input to the downstream neural network classifier.

In this study, we evaluate two types of commonly adopted neural network classifiers: 1) CNN and 2) CNN-LSTM. The architecture of the two types of classifiers is shown in Fig. 2. We use three convolutional layers, each followed by batch normalization, activation (ReLU), and max pooling. The output of the convolutional layers is flattened, followed by either LSTM and two fully connected layers, or three fully connected layers, to produce the prediction.

The CPI can sometimes be used as one dimension in the map, e.g., in a Range–Doppler–Time Cube, which can be fed into a 3DCNN classifier, or used as sequence timestamps, to act as input into LSTM. We use Table II to summarize possible data slicing, preprocessing, and downstream classifier

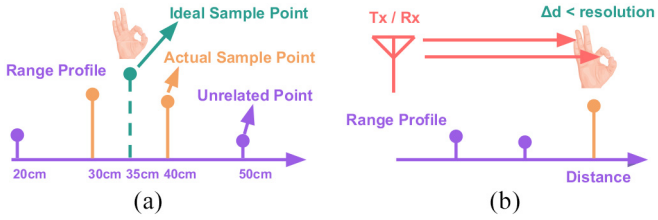


Fig. 3. Limitations of the conventional DFT preprocessing. (a) Fixed Fourier bases cannot best capture target information, and carry information not related to the target. (b) Resolution limitation makes it hard to capture fine-grained movements.

combinations. Note that since we are using FMCW radar, for Doppler–Time, Angle–Time, and Doppler–Angle–Time preprocessing, we still extract Range information first, and then estimate Doppler and/or AoA information for each Range bin, then sum along the Range axis. Some of the combinations have appeared in previous works, as we discuss in Section II, and in these works, the selection of the pipeline is largely based on the target application and radar configuration.

V. Learnable Preprocessing

A. Problems With Conventional DFT Preprocessing

For raw radar signals, the information of the targets is not encoded in certain data points, but rather hidden in the phase and magnitude variation of the raw signal. For example, the distances to the targets are encoded in the frequency components of a single chirp, and the Doppler and AoA information are hidden in the phase variation, either from time domain (chirps) or spatial domain (receiver array).

To expose such information, the preprocessing is often performed with conventional signal processing techniques, typically through DFT, as discussed before. The DFT can be viewed as a linear transformation of the raw radar data from the time/spatial domain to the frequency domain. Linear transformation can be viewed as change of basis. In DFT, the data is represented with equally spaced Fourier bases in the frequency domain, and the information corresponding to the targets is directly exposed. The preprocessed output can be largely regarded as 2-D/3-D images to be fed into downstream neural networks based on architectures widely explored in the computer vision area. Note that though DFT is a reversible operation, in previous pipelines, the phase information of the transformed data is lost before feeding into the downstream classifier because of the modulus operation.

Though conventional DFT preprocessing is simple, efficient, and straightforward, there are two issues which impact its utility.

Lack of Adaptive Transformation Bases: The conventional DFT is a universal solution for extracting frequency domain information and is not designed specifically for certain tasks. In the DFT, all the Fourier transformation bases are equally spaced and fixed, so the transformed data points might not best represent the target, as illustrated in Fig. 3(a). Note also the redundancy, as the part of the conventional DFT output that we are actually interested in only occupies a fraction of the whole preprocessing output. The noninformative part of the output may contain multipath reflections, noise, and environmental reflections. Sending this information into the downstream neural network will lead to a higher computational cost, and even worse, can make the neural network learn fake features and overfit to the training data. It is possible to have a specific DFT for each task, with different FFT lengths, nonuniform frequency spacings or frequency masks such that the domain transformation can focus more on relevant features corresponding to specific tasks. However, this means that we must choose specific parameters for each task, which relies on the experience of the developer and it is very time consuming to tune these hyperparameters.

Limited Resolution: Another problem is that while the DFT is complete and invertible, it has an inherent resolution limitation, which makes it hard for the DFT transformation output to represent the target properties on certain tasks, as demonstrated in Fig. 3(b). For example, the Range resolution of mmWave radar is typically > 5 cm, which makes it difficult to distinguish between two fingers in HGR. Also, the resolution limitations of Doppler and AoA make it hard to capture fine-grained movements, e.g., movements of two adjacent fingers. Note that such problem could not be simply fixed by "zero padding" in the time domain, as zero padding in time domain corresponds to interpolation in the frequency domain, which does not increase the resolution. This is because the resolution is a function of the raw signal bandwidth. Generally speaking, the resolution limitation is inherent to the transmit waveform. However, with additional prior knowledge, we are able to achieve better resolution, i.e., super-resolution techniques. For example, with the number of signal sources as a prior knowledge, multiple signal classification (MUSIC) algorithm can identify two targets that are not separable with DFT processing [32]. In our case, the neural network model is trained with training data, which inherently carries rich prior knowledge. As a result, with additional information, it can potentially overcome the resolution limitation of DFT-based processing.

B. CubeLearn Module Architecture

To deal with the above-mentioned limitations of conventional DFT preprocessing, in this work, we present the CubeLearn module to replace the DFT preprocessing, which can be trained together with the downstream classifier in an end-to-end way and aim to learn the best domain transformation for a specific task in a data-driven manner.

A neural network does not suffer from the limitations of conventional DFT. Through end-to-end training, the network is able to learn the transformation bases that best suit the target task. In addition, with prior knowledge acquired from training data, the neural network could be able to achieve better resolution. In fact, recent research has shown that super resolution of Range, Doppler, and AoA can be achieved through deep neural networks for raw radar signals [33], [34]. Furthermore, the preprocessing module could be trained to extract more informative features, rather than noise and environmental reflections, potentially carrying more useful information to the downstream classifier.

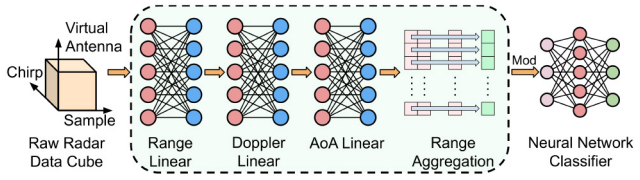


Fig. 4. CubeLearn architecture.

The architecture of our proposed CubeLearn module is shown in Fig. 4. The basic idea behind the design is to follow the conventional DFT processing method, as introduced in Section III, but in a learnable way. As the raw radar signal is complex valued, we use complex linear layers to directly consume the raw radar signal as input. Instead of processing all the information in the raw radar data cube in one layer, we use a stacked hierarchy of complex linear layers, each layer replacing a certain step in conventional DFT preprocessing (i.e., Range, Doppler, and AoA). This is because if we are to use a single complex layer, there would be too many parameters, making the neural network significantly more complex.

As we are using FMCW radar, the first step is to extract Range information from each chirp with a complex linear layer. This can be followed by other complex linear layers to extract Doppler and AoA information for different ranges. We can also aggregate the Doppler and AoA information at different ranges by calculating the sum along the Range axis. In conventional DFT preprocessing, the order of performing DFT on different dimensions of the input raw data cube does not affect the output. As a result, though we are following the conventional DFT preprocessing to organize the complex linear layers, the order of the linear layers could also potentially be swapped, which could be studied in future work.

To make the proposed CubeLearn module fairly act as a drop-in for the conventional DFT, we calculate the absolute (modulus) value before feeding the processed data into the downstream neural network classifier.

C. CubeLearn Module Weight Initialization

The DFT is a linear transformation from the time domain to the frequency domain. This can be represented by a complex linear layer with no bias in a neural network. We use DFT preprocessing as prior knowledge to initialize our proposed CubeLearn module, and let the neural network adjust the weights to adapt to a specific task through end-to-end training. The complex linear layer output can be represented as

$$\text{output}[k] = \sum_{m=0}^{M-1} w(k, m)\text{input}[m] \tag{1}$$

and we initialize the weights as

$$w(a, b) = e^{-j\frac{2\pi}{N}ab} \tag{2}$$

which exactly represents DFT. The bias of the complex linear layers is set to 0.

For Doppler and AoA transformation layers, we also need to shift the upper half and the lower half of the weights, since we want zero speed and zero AoA (i.e., in front of the sensor) initially to be in the middle of the learned map.

Note that though we initialize the complex layers as DFT, through end-to-end training, the transformation is very likely to no longer be Fourier Transform once trained.

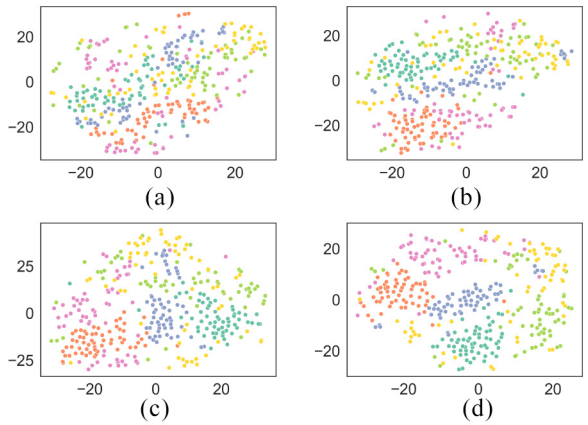


Fig. 5. t-SNE visualization of the extracted features. Different colors represent different gestures. (a) Conventional DFT. (b) CubeLearn, Epoch = 1. (c) CubeLearn, Epoch = 4. (d) CubeLearn, Epoch = 15.

D. Effect of CubeLearn Module

To better show the effects of our proposed CubeLearn module compared to the conventional DFT preprocessing and validate our assumptions, in this part, we demonstrate how the proposed CubeLearn module adaptively adjusts the weights, by visualizing the output of both the conventional DFT preprocessing module and the CubeLearn module through the training process. We use t-SNE, a widely adopted method for dimension reduction and data visualization [35]. Without loss of generality, we use D-T preprocessing + 2DCNN classifier model on HGR task, here, as an example. A detailed definition of the task as well as the description of the data set are introduced later in Section VI. There are 12 hand gestures in total, however, to make the figures more clear, we only visualize the preprocessed output of six gestures.

In Fig. 5, we first visualize the output of the conventional DFT preprocessing in Fig. 5(a), and then the change of the output of the proposed learnable D-T preprocessing module in Fig. 5(b)–(d). From Fig. 5(a), we can clearly see that, although in local regions, the output from certain classes appears to be denser, indicating that the conventional DFT extracts some common properties of the same class samples, on the whole, the features from different classes still tend to mix together. This shows that the conventional DFT preprocessing is not ideal for effectively extracting features from the raw radar signal. With the output from conventional DFT preprocessing, it could be quite challenging for the downstream classifier to produce accurate predictions.

On the other hand, we can see from Fig. 5(b)–(d), with the use of the CubeLearn module, the extracted features of different classes become more distinguishable as the training proceeds. This shows that during end-to-end training, the optimizer is able to adjust the weights in the CubeLearn module

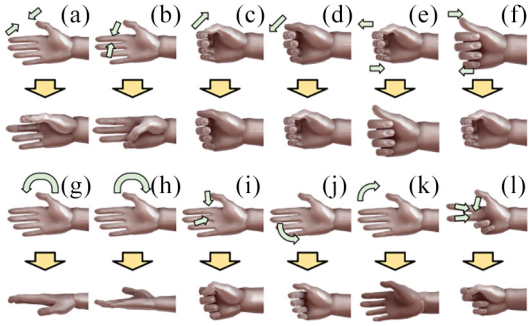


Fig. 6. Hand gesture set design.

to produce more informative outputs. As the outputs of the preprocessing module become more distinguishable, it greatly reduces the burden of the downstream classifier and makes the task of achieving a higher accuracy simpler. In later sections, we provide a further evaluation and discussion of the strengths and limitations of the proposed CubeLearn module.

VI. Experiment Setup

We conduct extensive experiments on three most commonly seen interaction tasks with mmWave radar, including HGR, arm gesture recognition (AGR), and HAR. As there is no existing public data set that contains raw complex radar data available for the above tasks, we collected our own data set for evaluation. We try to follow existing publications on experiment settings and gesture/activity set designs. To understand the evaluation results, we provide the design of tasks, hardware settings, and software implementation details in this section.

A. Experiment Design

In this part, we introduce our design of the hand gesture set, arm gesture set, and activity set in our experiment.

1) Task 1 (Hand Gesture Recognition): For HGR, we follow the gesture set design of several previous works [3], [16], [20] and define our gesture set used in this work. Our gesture set design is shown in Fig. 6. The hand gesture set contains: (a) Pinch Index; (b) Pinch Pinky; (c) Slide Right; (d) Slide Left; (e) Rub Forward; (f) Rub Backward; (g) Rotate Left; (h) Rotate Right; (i) Fist; (j) Swipe Left; (k) Swipe Right; and (l) Zoom Out. The palm is facing the sensor at the beginning of each gesture, and the gestures are based on the right hand and are performed 20 cm away in front of the sensor, as shown in Fig. 8(b).

Note that our hand gesture set design is much more difficult than the hand gesture sets in most of the previous works on HGR with mmWave radar, as our gesture set only involves hand movement, mostly fingers, and does not include forearm movement.

2) Task 2 (Arm Gesture Recognition): We design 12 commonly used gestures based on several previous works on radar AGR [2], [36], as shown in Fig. 7. The gestures only involve right arm gestures and are performed 1.5 m away in front of the radar, as shown in Fig. 8(c). The arm gesture set contains: (a) Side Lift; (b) Side Down; (c) Front Lift; (d) Front Down; (e) Forearm Lift; (f) Forearm Down; (g) Push; (h) Pull; (i) Swipe Right; (j) Swipe Left; (k) Diagonal Lift; and (l) Diagonal Down.

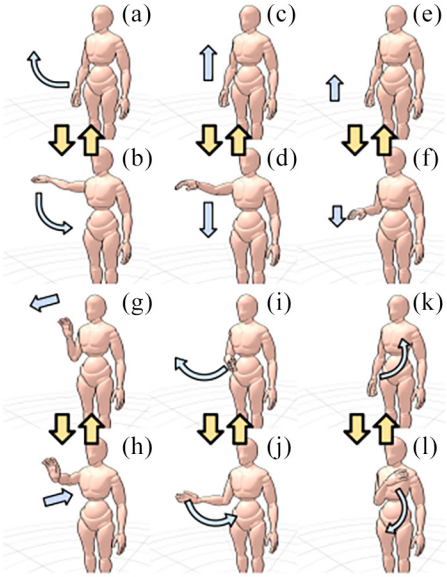


Fig. 7. Arm gesture set design.

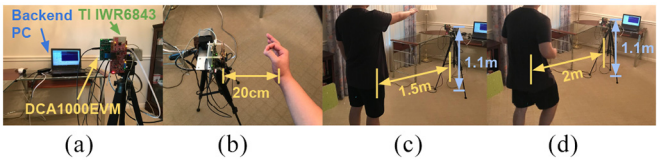


Fig. 8. Experiment settings. (a) Device setup. (b) Hand gesture data collection. (c) Arm gesture data collection. (d) Activity data collection.

3) Task 3 (Human Activity Recognition): Another commonly seen application is HAR. Different from HGR and AGR, HAR usually involves continuous movements, which means the activities do not have a certain "start time" and "end time." Instead of trying to identify each start time as we do in gesture recognition, we simply cut the data into equal-length samples. We also follow the design in previous works [6], [7], [18], excluding those we believe that are not appropriate for participants to perform (e.g., crawling) or those that are too physically demanding (e.g., jumping) for typical smart-home applications. We include the following six activities in our data set: 1) marching on the spot; 2) jogging on the spot; 3) clapping; 4) waving right hand; 5) sweeping the floor with the right hand; and 6) rubbing left arm with right hand. The activities are performed 2 m away in front of the radar in our data collection, as shown in Fig. 8(d).

B. Data Collection Setup

In this article, we use a commercial-off-the-shelf Texas Instruments IWR6843ISK mmWave sensor carried by MMWAVEICBOOST and DCA1000EVM for raw data streaming. The sensor is mounted on a tripod, with a height of around 110 cm, as shown in Fig. 8(a). Raw radar data is streamed and stored in PC continuously. For hand gesture and AGR, the starting time of each action is logged and the data

corresponding to each action is later extracted from the raw data file according to the starting time with a length of 1 s (10 CPIs).

We configured the radar with the following parameters: the starting frequency of each chirp is configured to 60.25 GHz with a frequency slope of 60 GHz/ms; the sampling rate is set to 10^7 samples per second and we take 256 ADC samples at each receiver antenna during a chirp; the period of each CPI is 100 ms, containing 128 chirp loops where the three transmitters are activated one by one.

We invite eight participants for data collection, including four males and four females, aged 26–55, with height approximately from 160 to 175 cm, and bodyweight approximately from 45 to 90 kg. For each participant, we collected 30 samples per hand/arm gesture, and 1 min of activity, which is also cut into 30 samples. The background environment of data collection could vary across participants, but the experiment settings are the same. The data from the eight participants are divided into two parts: six people are used as "in-set" users, whose data is used for training and in-set testing; the remaining two people are "out-of-set" users, whose data is used to test the generalization abilities of the trained model. For in-set users, 15 of the 30 samples of each gesture/activity are used for training, five used for validation and the remaining ten used for testing, while for out-of-set users, all 30 samples of each gesture/activity are used for testing.

C. Neural Network Implementation

We implement the neural networks with PyTorch. The complex layers are implemented based on an open-source Python module called CplxModule.[3] For Range and Doppler transformations, we configure the output size of the linear layer to be the same as the input size. We only use azimuth AoA information in this work, and for extracting AoA information, we configured the output size to 64 instead of the input size 8, so that the preprocessed data can be better fed into the convolutional layer for further feature extraction. For D-A-T and R-D-A-T preprocessing, we use the first half of the raw data on sample and chirp axes because of GPU memory limitation. This does not lose much information on Range and Doppler dimensions, as we will discuss in Section X.

For the implementation of the neural network classifiers, we use the same settings for each model. The kernel size for the convolutional layers are set to 3 and the channels are set to 4, 8, and 16, respectively. We apply max-pooling operation of size 2 on each dimension after each convolutional layer excluding the CPI dimension. The LSTM is set to have a hidden size of 512. The output size of the three fully connected layers is set to 512, 128, and 12/6 (depending on the number of classes), respectively.

We use the cross-entropy loss to train the network. The neural networks are trained with Adam optimizer. Each model is trained for 30 epochs, and the best weights are saved according to validation accuracy and loss (accuracy is prioritized than loss).

[3]https://github.com/ivannz/cplxmodule

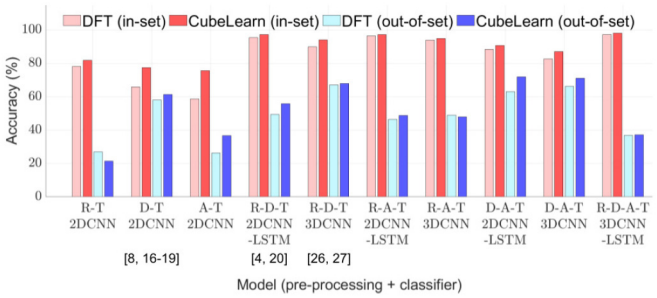


Fig. 9. HGR accuracy.

VII. Evaluation

In this section, we first evaluate the accuracy of DFT preprocessing-based pipelines and CubeLearn-based pipelines on HGR task, then extend to AGR and HAR tasks. Some of the pipelines with conventional DFT preprocessing are widely adopted by previous works on related topics, e.g., D-T preprocessing + CNN classifier [7], [15], [16], [17], [18], R-D-T preprocessing + CNN-LSTM classifier [25], [26], etc. As a result, they also serve as a comparison to previous related works.

We also evaluate the trained models on the out-of-set users whose data is not included in the training. Generalization accuracy tends to be lower, as recognition accuracy can be impacted by the shape and size of the user's hand/body, how the user performs the action, as well as the environmental noise. This situation can certainly be mitigated by collecting more training data with various users and environments, or adopt transfer learning and life-long learning techniques. As our main aim, here, is to evaluate the generalization ability of our proposed CubeLearn-based methods against conventional DFT preprocessing-based pipelines, we do not focus on improving the generalization ability of a specific model/task here.

Besides validating the performance of our proposed CubeLearn module, this section also serves as a comparison between preprocessing and classifier combinations on different tasks.

A. Hand Gesture Recognition Evaluation

We first evaluate our proposed CubeLearn module against conventional DFT on HGR task. The results are shown in Fig. 9.

1) Impact of CubeLearn Module: We can see from the figure that, with CubeLearn, the model is able to have consistently better or at least comparable performance than the DFT preprocessing counterparts. This is, especially, pronounced on simpler models, e.g., R-T preprocessing + 2DCNN classifier, A-T preprocessing + 2DCNN classifier, and the widely adopted D-T preprocessing + 2DCNN classifier, with over 10% of accuracy improvement for the latter two models. Even for some of the models that already have a reasonably high recognition accuracy ($\geq$ 90%) with conventional DFT preprocessing, our proposed CubeLearn module is usually able to further improve attained accuracies by around 1%–5%. These

results show that our proposed CubeLearn module is a powerful tool to boost recognition accuracy on motion recognition tasks with mmWave radar.

2) Impact of the Combination of Preprocessing and Classifier: For in-set testing, we generally have better performance with higher dimensional preprocessing. The R-D-A-T preprocessing-based model achieves the best testing accuracy of 97.36% with conventional DFT and 98.06% with CubeLearn. And also, 2-D preprocessing (R-D-T, R-A-T, and D-A-T) have significantly better performance than 1-D preprocessing (R-T, D-T, and A-T). Higher dimensional preprocessing inherently has more informative features, and the additional information can be utilized to better distinguish between different hand gestures. Also for R-D-T, R-A-T, and D-A-T preprocessing, we find that the 2DCNN-LSTM classifier behaves better than the 3DCNN classifier, probably because the LSTM can better track the temporal relationships between frames in a gesture sample through the explicit latent feature space.

3) Generalization to Out-of-Set Users: When encountering out-of-set users, our proposed CubeLearn can still achieve better performances than DFT preprocessing for the majority of the models.

Having a closer investigation we can observe that the CubeLearn tends to further increase the generalization ability on more generalizable models, as these models capture more features that represent the underlying physical model, rather than some special properties of the training data. For example, in the D-A-T + 2DCNN-LSTM classifier, with DFT preprocessing module, the model is able to have around 63% of testing accuracy on out-of-set subjects, and the proposed CubeLearn module can improve the accuracy to around 72%.

Comparing across different models, we find that the models with Doppler information have better generalization abilities. This is because Doppler information is invariant to the location where the gesture is performed, as long as the gesture or activity is oriented toward the radar. As CubeLearn can extract more distinguishable Doppler features through a learnable complex linear layer, the generalization abilities of Doppler-based models tend to further improve compared to conventional DFT preprocessing counterparts.

However, with the strong feature extracting ability of the CubeLearn module, some specific characteristics of the training samples could also be extracted and carried over, which sometimes makes it harder for the model to generalize to previously unseen subjects. For example for R-T preprocessing + 2DCNN classifier, although CubeLearn can attain better test performance with in-set users, it generalizes less well than conventional DFT preprocessing, as the Range feature itself is inherently not able to generalize well on out-of-set subjects, as the conventional DFT-based preprocessing model achieving less than 30% accuracy. Luckily when solving real-world problems, we would most likely pick models that use features with strong generalization ability, and in these cases, the CubeLearn module could also be beneficial to the testing accuracy on out-of-set users.

With respect to radar data cube slicing, different from in-set testing, higher dimensional preprocessing does not always lead to better generalization performance. For example, the D-A-T + 3DCNN model has better generalization ability than the R-D-A-T + 3DCNN-LSTM model, also because the R-D-A-T features overfit to the training subjects. This could also be a function of the training set size—a data set of hundreds of users would likely generalize well, even for higher dimensional models.

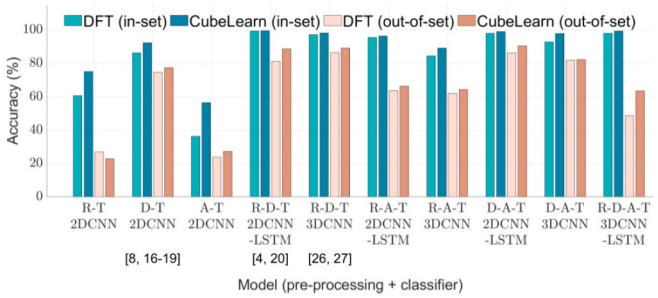


Fig. 10. AGR accuracy.

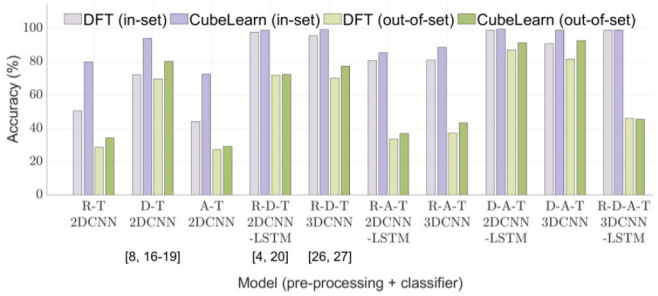


Fig. 11. HAR accuracy.

B. Generalization to AGR and HAR Tasks

We further evaluate our proposed method against conventional DFT on the other two tasks, and the results are shown in Figs. 10 and 11. The AGR task is similar to the HGR task, where each gesture is represented by a sequence of frames, but it is much easier as the movements are bigger which makes them more obvious to mmWave radar. HAR is fundamentally different as it mainly involves recognizing continuous and repetitive movements.

From the in-set testing result, we can see that the model performances on AGR and HAR are generally better than the HGR task. Still, our CubeLearn module can consistently improve the performance of different pipelines, by extracting the most informative features from the raw radar signal cube. The proposed module can be extremely beneficial sometimes, e.g., increasing the accuracies of some of the simpler models (R-T, D-T, and A-T) by almost 15%–20%. This is an important finding, as R-T and D-T are the most basic signals which all FMCW radars will support, regardless of the number of antennas, and they are more computational efficient for edge devices. This means that a simpler radar (with respect to hardware configuration) could achieve the performance of a much more sophisticated device, merely through the use of CubeLearn.

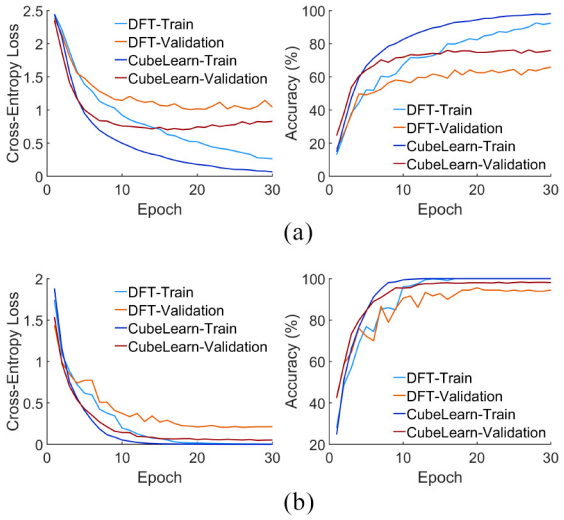


Fig. 12. Visualization of the training processes. (a) HGR: D-T preprocessing + 2DCNN classifier. (b) HAR: R-D-T preprocessing + 3DCNN classifier.

Different from HGR/AGR case, in the HAR task, we find that using LSTM in the downstream classifier does not always yield better performance. For example when using R-A-T preprocessing, the 3DCNN classifier has better performance than the 2DCNN-LSTM classifier. This suggests that the LSTM might be better at recognizing gestures than continuous motion, and for continuous motion recognition, both classifiers are worth trying.

The accuracies for out-of-set test are also higher as the tasks themselves are less challenging to generalize due to their more pronounced motion. The accuracies of some models on out-of-set subjects are comparable to the accuracies on in-set subjects, such D-T + 2DCNN in HAR task. By analyzing the out-of-set test results, we can arrive at a similar conclusion as in the HGR task, i.e., our proposed CubeLearn module can be beneficial for most models, and Doppler features are more important for motion recognition than Range or AoA features.

C. Convergence Analysis

We further compare the convergence speed of methods based on DFT preprocessing and CubeLearn. We visualize the training process, including the training accuracy/loss and the validation accuracy/loss, for D-T preprocessing + 2DCNN classifier on HGR task and R-D-T preprocessing + 3DCNN classifier on HAR task in Fig. 12. In both cases, the model with CubeLearn is able to converge in fewer epochs, typically within 15 epochs, with loss decreasing and accuracy increasing much faster. Furthermore, from Fig. 12(b) we can see that, though the models converge to approximately the same training loss and accuracy, completely fitting the training set, there is a constant validation loss/accuracy difference between the models, which shows the network with the CubeLearn module can actually better learn the physical model behind the task. Note that this shows that the superior performance of the CubeLearn module is not merely because of more trainable parameters, as in this case, CubeLearn-based model and conventional DFT-based model both completely fit the training data, while the CubeLearn-based model still yields better performance on the validation/testing set.

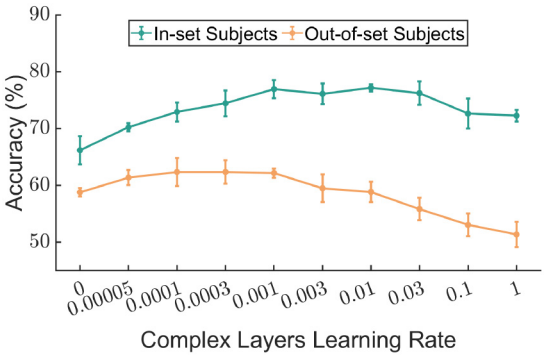


Fig. 13. Test accuracy versus different learning rates.

VIII. Ablation Study

A. Impact of Complex Layer Learning Rate

The learning rate of the CubeLearn module controls the speed at which the complex layer weights vary, which can be set differently from the learning rate of the neural network classifier. With a higher CubeLearn module learning rate, the weights of the complex linear layers change faster during training—the coupling between the preprocessing and classification networks could cause the system as a whole to converge to a suboptimal set of weights. To find the optimal learning rate for the complex layers (or the ratio between the learning rates of the CubeLearn module and the classifier), in this part, we study the impact of the CubeLearn module learning rate on the model overall accuracy. We use the D-T preprocessing + 2DCNN classifier model on HGR task, here, for demonstration. The learning rate of the neural network classifier is set to 0.0003 for all the cases. Each model runs multiple times. The result is shown in Fig. 13.

From the results, we can see that, both in-set and out-of-set testing, with the increase of the CubeLearn module learning rate, the model accuracy first increases, then decreases. Note that when the learning rate is 0, the model degenerates to a conventional fixed DFT preprocessing + 2DCNN classifier model. With the learning rate from 0.00005 to 0.0001, the optimizer is able to adjust the weights of the CubeLearn module to achieve better model accuracy, for in-set and out-of-set subjects test. With the learning rate further increasing, the model is likely to overfit to the training subjects, and even to the training samples. As shown in the figure, the generalization accuracy begins to decrease when the learning rate of the CubeLearn module is larger than 0.001, which shows that the model is overfitted to the training subjects. The in-set test accuracy is able to be maintained at a relatively high level until the learning rate reaches 0.03. When the learning rate is larger than 0.03, the model further overfits to the training samples, leading to an accuracy degradation on the in-set test as well.

Fig. 14 shows the training and validation loss and accuracy curve of the models with different CubeLearn module learning

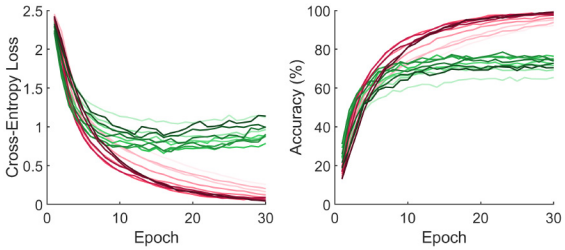


Fig. 14. Loss and accuracy curve. Red: training; Green: validation. Darker color represents higher CubeLearn module learning rate.

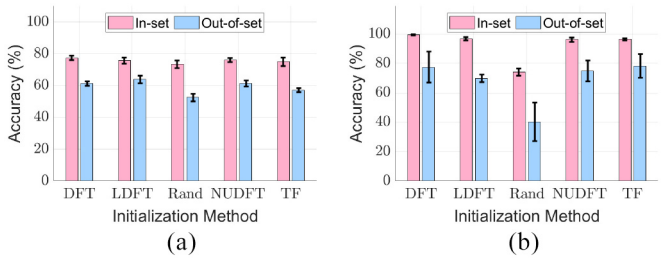


Fig. 15. Initialization method ablation study. (a) D-T + 2DCNN classifier. (b) R-D-T + 3DCNN classifier.

rates. We can see that for training set, with a higher learning rate the model is able to converge at a smaller training loss, and larger training accuracy. However, a suitable learning rate could make the model converge faster. For the validation set, either a learning rate that is too small or too large yields suboptimal performance, and the smallest loss and highest accuracy can only be achieved with a proper learning rate of the complex layers, which is 0.001 in this case. Note that the optimal learning rate ratio between the CubeLearn module and the neural network classifier can vary for different preprocessing and classifier combinations as well as on different tasks.

B. Impact of Weight Initialization Method

The initialization of the weights in the stacked complex linear layers affects the model accuracy as well as convergence. In this part, we try different weight initialization strategies, including: 1) log-DFT (LDFT) initialization, where the weights are initialized with Fourier Transform bases more focusing on the low-frequency part; 2) default random initialization (He Initialization [37]) for both real and imaginary parts of the complex weights (**Rand**); 3) Random nonuniform DFT (NUDFT) bases initialization, where the initialize weights are based on random Fourier Transform bases (sorted) (NUDFT); and 4) target focus initialization, where the information (e.g., distance, velocity, and AoA) regarding the target is known and the bases of the Fourier initialization are generated close to the target frequency with normal distribution (TF).

We compare the complex linear layers initialization methods with two most common models: 1) D-T preprocessing + 2DCNN classifier on HGR task and 2) R-D-T preprocessing + 3DCNN classifier on HAR task. The evaluation result is shown in Fig. 15, where we can see that the model generally is not very sensitive to the weight initialization method and is able to converge in each case.

LDFT initialization focuses on the low-frequency part, which produces slightly better generalization accuracy for the HGR task, because in the Range domain, the high frequencies correspond to distant targets, which is the background environment in this case. However, we observe a slight performance downgrade for the HAR task, because the user is 2 m away from the sensor in the HAR task, which is approximately in the middle of the frequency spectrum rather than the low-frequency part. With the LDFT initialization focusing on the low-frequency part, there are in fact fewer bases initially focus on the target. As a result, LDFT might be beneficial for recognizing targets that are close to the sensor, however, this type of initialization needs careful design.

We find that with random initialization, the model has worse in-set and out-of-set accuracy. Especially for the HAR case, the accuracy drops to below 80% and the out-of-set generalization accuracy even drops to below 40%. For HGR, the out-of-set generalization accuracy also drops significantly. Using random initialization basically means giving up the prior knowledge about radar signal processing, which is not desirable, especially, when the number of samples in the training set is limited.

The Random NDUFT initialization is theoretically quite similar to DFT bases initialization, for the bases are sampled with uniform distribution across the frequency spectrum and reorganized from low frequency to high frequency. However, from the result, we see that this kind of initialization is not as stable as uniform DFT bases initialization, with slightly worse average performance.

Target focus initialization is very hard to use in practice, since it requires prior knowledge of the location, velocity, and AoA of the target, and a lot of engineering effort. Interestingly, we do not observe better performance even with careful parameter tuning.

In conclusion, in practice, it is reasonable to simply use DFT bases to initialize the weights of the complex linear layers due to their simplicity, high accuracy, and stable performance.

C. Impact of Network Structure

We further conduct an ablation study of the preprocessing module structure. We also use the D-T preprocessing + 2DCNN classifier model on HGR task and R-D-T preprocessing + 3DCNN classifier model on HAR task here, as they are the most commonly seen models in previous works. Besides, learnable D-T preprocessing can be directly compared to RadarNet [12] and learnable R-D-T preprocessing can be directly compared to 2-D Sinc/Wavelet Filters [11]. As mentioned in Section II, RadarNet and 2-D Sinc/Wavelet Filters are two previously proposed methods for end-to-end radar signal processing.

We compare the performance of our proposed structure to the following variations: 1) Adding activation between complex linear layers (**ModReLU** [38], $\mathbb{C}$**ReLU** [39], and **zReLU** [40]); 2) learnable NUDFT, where we constrain the learned weights to still represent Fourier Transform bases;

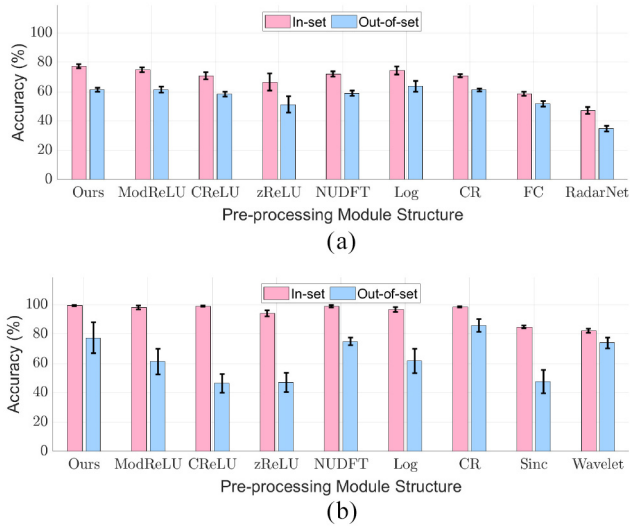


Fig. 16. Structure ablation study. (a) HGR: D-T + 2DCNN classifier. (b) HAR: R-D-T + 3DCNN classifier.

3) log after modulus (**Log**), where the log operation is applied after taking the modulus to represent power spectrum as in [12] and [41]; 4) clutter removal (CR), where we subtract the chirp average from the raw radar data cube before feeding it into the network; 5) fully complex network (FC), where we do not take modulus after preprocessing, and use a fully complex version of the downstream classifier; 6) **RadarNet** [12]; and 7) 2-D **Sinc** Filters and 2-D **Wavelet** Filters [11].

As RadarNet [12] is designed for CW radar rather than FMCW radar, we first do Range DFT on the raw radar data, and then apply the preprocessing module introduced in RadarNet for Doppler estimation on each Range bin, then sum along the Range axis. Besides, we use log operation instead of tanh which is suggested in the original paper, as the network does not converge when using tanh in our case. For Sinc and Wavelet Filters [11], we apply it on each CPI and use the CPI number as an additional dimension, since the radar configuration is different in this work. Besides, we find that the hyperparameters greatly impact the Sinc/Wavelet Filter performance, so we try different combinations and select the best one. Both RadarNet and Sinc/Wavelet Filters takes real-valued data, and we disregard the imaginary part of the data.

The result is shown in Fig. 16. We can see that adding an activation function between the complex linear layers decreases the model performance, especially, the generalization ability. This is probably because in the stacked complex linear layers, extracting the Doppler information through the transformation of the latter complex linear layer relies on the output phase information of the previous complex linear layer. Adding a nonlinear activation between the complex linear layers, especially, $\mathbb{C}$ReLU and zReLU, could result in the valuable phase information being altered or lost.

NUDFT structure produces a slightly worse result than CubeLearn, since the change of the weights is restricted and not as flexible. As for adding the log operation after the CubeLearn module, we do not observe performance improvement. In fact, for the HAR case, adding the log operation downgrades the generalization ability, probably because the difference of peak magnitude and average magnitude is smaller, making the learned features less distinguishable.

CR operation removes reflection of static targets (e.g., background) and makes the model learn properties of the moving target, which increases the model generalization ability in HAR case. On the other hand, as it actually cancels out all the static information, including the information of the target, for HGR task where the hand is very close to the sensor and occupying a larger field of view, removing the static information slightly decreases the model performance. Besides, if we are to recognize the static target, e.g., in static gesture recognition, CR will certainly not work. As a result, whether adding CR is beneficial or not depends on the task. For tasks to recognize moving targets that are relatively further away, adding CR could increase the model generalization ability by removing the background reflections. While recognizing partial static or static targets, especially, in cases where the target occupies a large field of view and the reflection from the environment has limited impact, adding CR could possibly decrease the performance.

Using the processing module in RadarNet lowers the accuracy. This is because RadarNet only consumes the real input, so it loses half of the information, which leads to a worse accuracy. 2-D Sinc Filters and 2-D Wavelet Filters also take real-valued input, so the accuracy is not as good as other structures. We find that compared to our proposed CubeLearn with complex linear layers, 2-D Sinc Filters and 2-D Wavelet Filters are very easy to become overfitted to the training samples, and are very sensitive to the layer parameters. For out-of-set test, we observe that 2-D Wavelet Filters have competent generalization abilities, much higher than 2-D Sinc Filters.

In conclusion, compared to other variations, our proposed stacked complex linear layers achieve good and stable performance. Adding CR could possibly improve the generalization abilities of the model in some tasks, but needs to be used with caution.

IX. Running Time Analysis

The DFT is usually implemented with the Fast Fourier Transform algorithm, which is very computationally efficient ($\mathcal{O}(n \log n)$). Using the stacked complex linear layers introduces more parameters in the network, which will result in larger complexity and longer inference time. In this part, we analyze the running time of the proposed method against classical hybrid baselines on PC and Raspberry Pi. The experiment PC we use features an AMD Ryzen 3800X CPU with 64-GB memory, and Nvidia RTX2080Ti graphics card, and the Raspberry Pi we use is Raspberry Pi 4 with 8-GB main memory. For DFT-based preprocessing, we use "torch.fft" in "pytorch" package for possible GPU acceleration when testing on PC, and use "scipy.fftpack.fft" in "scipy" package on Raspberry Pi since torch.fft is not supported on ARM architecture. The structures of the neural networks are discussed in Section VI-C. Note that the running time reported in this section is based on models for HGR/AGR task.

TABLE III
RUNNING TIME ON PC AND RASPBERRY PI PER SAMPLE

Model	PC Time (ms)		Raspberry Pi Time (ms)	
	DFT	CubeLearn	DFT	CubeLearn
R-T + 2DCNN	0.99	1.25	15.00	35.71
D-T + 2DCNN	1.05	1.65	77.41	484.13
A-T + 2DCNN	1.08	1.61	39.51	165.32
R-D-T + 2DCNN-LSTM	1.63	2.27	625.72	1018.70
R-D-T + 3DCNN	1.14	1.75	419.31	745.10
R-A-T + 2DCNN-LSTM	1.62	2.21	308.38	528.02
R-A-T + 3DCNN	1.15	1.69	222.17	357.00
D-A-T + 2DCNN-LSTM	3.18	5.29	1486.26	2456.30
D-A-T + 3DCNN	2.92	5.02	1626.13	2513.14
R-D-A-T + 3DCNN-LSTM	18.73	24.38	-	-

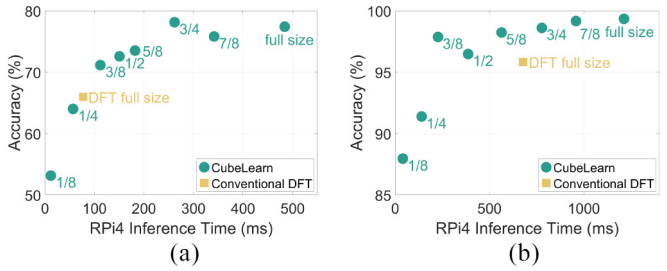


Fig. 17. Accuracy and inference time with different sizes of raw data input. (a) HGR: D-T + 2DCNN classifier. (b) HAR: R-D-T + 3DCNN classifier.

The result is summarized in Table III. Not surprisingly, the computation of complex linear layers is more expensive than FFT, as each layer is actually called multiple times during inference. With GPU acceleration on PC, adding the learnable complex layers would add 47.7% of inference time for one sample on average (SD = 15%). When running with the ARM CPU on Raspberry Pi 4, the running time for each sample increases 152.7% on average (SD = 163.1%). The computational cost grows exponentially with the number of dimensions of the raw data cube used. For example, with D-A-T or R-D-A-T preprocessing, the neural network has to perform complex linear transformation on all three dimensions, which leads to a large increase in inference time. However, in practice, complex models (e.g., D-A-T and R-D-A-T preprocessing-based models) are less likely to be used on edge devices. For simpler models that can run real time on edge devices, CubeLearn could greatly improve the recognition accuracy, as shown in previous sections, with acceptable complexity increase. Furthermore, we can have a smaller input size with our proposed end-to-end network, to achieve better performance and even smaller running time than DFT baselines, as introduced in the next section.

X. IMPACT OF RAW DATA SIZE

The raw radar signal contains repetitive patterns on ADC samples in a chirp, and across chirps in a CPI. As a result, lowering the number of samples and chirp number in a CPI would possibly still yield a similar level of recognition performance while making the neural network much more lightweight. In this part, we tested the performance of two models with various input sizes: D-T preprocessing + 2DCNN classifier on the HGR task and R-D-T preprocessing + 3DCNN classifier on the HAR task, as well as the running speed on Raspberry Pi4. Note that the running time of the R-D-T preprocessing + 3DCNN classifier is different from the one reported in Section IX because each sample lasts 2 s for HAR task, instead of 1 s for HGR task which we studied previously.

The input, here, is actually a series of sample-chirp data slices. For both axes, we try using the first $1/8, 1/4, \ldots, 7/8$ of the data points on each dimension as input. As we have a smaller size of the input, the size of the preprocessing module and the neural network classifier is also smaller, which reduces the computational complexity. The result is shown in Fig. 17, the text beside each data point refers to the ratio of the samples and chirps we use for each CPI. To have a fair comparison, each model is run multiple times.

We can see that the full-sized input actually contains redundant information. With even 3/8 of the original input size on sample and chirp dimension, we are still able to achieve relatively high performance, better than full-sized input with conventional DFT preprocessing. For the HAR case, with 3/8, 1/2, or 5/8 size on each dimension of the original input, the inference time on Raspberry Pi 4 of the CubeLearn-based model is smaller than conventional DFT preprocessing-based model with full-sized input, but achieves better classification accuracy. Especially with 3/8 of size on both dimensions, the accuracy could reach 98% while the inference time is approximately 200 ms.

In fact, for specific tasks, we can further reduce the complexity and running time by, e.g., using fewer neurons to represent the transformation output with the initialization of only the Fourier bases corresponding to the distance and velocity of the target. Besides, we can also utilize hardware-level parallel, processing the data of the current CPI while receiving the data for the next CPI. As a result, we believe that with an optimized implementation as future work, our proposed end-to-end model would execute in real time on resource-constrained devices, with superior performance to the DFT front-end.

XI. DISCUSSION

A. Model Selection

As we have discussed in Section VII, R-D-A-T preprocessing + 3DCNN-LSTM classifier shows good accuracy on different motion recognition tasks, especially, with our proposed learnable preprocessing module, as information from all three dimensions (Range, Doppler, and AoA) are utilized for target classification. However, it mainly has two problems which limit its adoption. The first one is computational complexity, which can be higher especially, on edge devices without GPU acceleration (an issue similarly shared by D-A-T preprocessing). Another problem is that it requires multiple receivers for AoA estimation, which is not always supported by the hardware itself, so this type of model is best suitable for MIMO radar with abundant computational resource.

Doppler information is the most important for motion recognition. As a result, considering the complexity and accuracy on both in-set users and out-of-set users, D-T or R-D-T preprocessing can be good choices, especially, for 1Tx–1Rx radar configuration. Besides, D-T preprocessing can be extended to other types of radar as well, e.g., CW radar.

B. Robustness

For learning-based methods, including our proposed neural network in this article, the generalization ability, i.e., robustness to different situations, is always an important concern. In this article, we make a preliminary investigation into this topic by collecting data from different users and in different environments. We suggest that more comprehensive research on this topic can be done in future works.

C. Utilizing Elevation Angle Information

In this article, we evaluated and compared models that are based on Range, Doppler, and Azimuth Angle information as inputs. With a 2-D MIMO antenna array, the radar is also able to capture elevation angle information, which can potentially be used for human gesture/activity recognition as well. However, in this case, the network design could be different. If we try to utilize all the information simultaneously, i.e., Range, Doppler, azimuth angle, and elevation angle, then we would need to apply a 4-D convolutional layer which is not commonly seen in neural network architectures. Alternatively, we can explore other network structures.

D. Other Structures

While the order of the linear layers is worth studying, other structures for extracting information from raw radar signals are also worth exploring. The transformer architecture [42] was proposed in recent years, which has shown very good performance on a wide variety of tasks like machine translation. With positional encoding for different virtual antennas, we could use the encoder part of the transformer model, potentially a complex-weighted version, to extract Range, Doppler, and AoA features together into a latent embedding and directly perform classification on the latent vector. In this way, we can get rid of the downstream CNN/CNN-LSTM architecture as well. We believe this is also a promising direction to study in future work.

XII. Conclusion

In this article, we propose CubeLearn, a learnable preprocessing module to replace conventional DFT preprocessing in mmWave FMCW radar gesture/activity recognition pipelines. We demonstrate that through end-to-end training, our proposed CubeLearn module is able to extract more distinguishable features than conventional DFT preprocessing, which makes it easier for the downstream classifier to make correct predictions and improve the overall model accuracy. We evaluate our proposed method on our own collected data set of hand gestures, arm gestures, and activities. Results show that for all the tasks and different preprocessing/classifier combinations, the classification accuracy can be consistently improved with the proposed CubeLearn module, and the training is able to converge in fewer epochs. We envision our proposed method as a small step toward a universal feature extractor for end-to-end deep learning on raw mmWave radar data. Future works would be focusing on exploring more efficient end-to-end structures and improving the model robustness.

References

[1] A. Singh, S. U. Rehman, S. Yongchareon, and P. H. J. Chong, "Multi-resident non-contact vital sign monitoring using radar: A review," *IEEE Sensors J.*, vol. 21, no. 4, pp. 4061–4084, Feb. 2021.

[2] S. Palipana, D. Salami, L. A. Leiva, and S. Sigg, "Pantomime: Mid-air gesture recognition with sparse millimeter-wave radar point clouds," *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 5, no. 1, pp. 1–27, 2021.

[3] S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges, "Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum," in *Proc. 29th Annu. Symp. User Interface Softw. Technol.*, 2016, pp. 851–860.

[4] F. Jin, A. Sengupta, and S. Cao, "mmFall: Fall detection using 4-D mmWave radar and a hybrid variational RNN AutoEncoder," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 2, pp. 1245–1257, Apr. 2022.

[5] B. Wang, L. Guo, H. Zhang, and Y.-X. Guo, "A millimetre-wave radar-based fall detection method using line kernel convolutional neural network," *IEEE Sensors J.*, vol. 20, no. 22, pp. 13364–13370, Nov. 2020.

[6] 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 *Proc. 3rd ACM Workshop Millimeter Wave Netw. Sens. Syst.*, 2019, pp. 51–56.

[7] K. Ahuja, Y. Jiang, M. Goel, and C. Harrison, "Vid2Doppler: Synthesizing doppler radar data from videos for training privacy-preserving activity recognition," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2021, pp. 1–10.

[8] A. Khamis, B. Kusy, C. T. Chou, M.-L. McLaws, and W. Hu, "RFWash: A weakly supervised tracking of hand hygiene technique," in *Proc. 18th Conf. Embedded Netw. Sensor Syst.*, 2020, pp. 572–584.

[9] S. Ahmed, K. D. Kallu, S. Ahmed, and S. H. Cho, "Hand gestures recognition using radar sensors for human–computer–interaction: A review," *Remote Sens.*, vol. 13, no. 3, p. 527, 2021.

[10] X. Li, Y. He, and X. Jing, "A survey of deep learning-based human activity recognition in radar," *Remote Sens.*, vol. 11, no. 9, p. 1068, 2019.

[11] T. Stadelmayer, A. Santra, R. Weigel, and F. Lurz, "Data-driven radar processing using a parametric convolutional neural network for human activity classification," *IEEE Sensors J.*, vol. 21, no. 17, pp. 19529–19540, Sep. 2021.

[12] W. Ye, H. Chen, and B. Li, "Using an end-to-end convolutional network on radar signal for human activity classification," *IEEE Sensors J.*, vol. 19, no. 24, pp. 12244–12252, Dec. 2019.

[13] R. Zhao, X. Ma, X. Liu, and J. Liu, "An end-to-end network for continuous human motion recognition via radar radios," *IEEE Sensors J.*, vol. 21, no. 5, pp. 6487–6496, Mar. 2021.

[14] X. Zheng, Z. Yang, K. He, and H. Liu, "Hand gesture recognition based on range doppler-angle trajectory and LSTM network using an MIMO radar," in *Proc. 11th Int. Conf. Signal Process. Syst.*, vol. 11384, 2019, Art. no. 113840P. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11384/2558317/Hand-gesture-recognition-based-on-range-Doppler-angle-trajectory-and/10.1117/12.2558317.short

[15] B. Dekker, S. Jacobs, A. Kossen, M. Kruithof, A. Huizing, and M. Geurts, "Gesture recognition with a low power FMCW radar and a deep convolutional neural network," in *Proc. Eur. Radar Conf. (EURAD)*, 2017, pp. 163–166.

[16] X. Zhang, Q. Wu, and D. Zhao, "Dynamic hand gesture recognition using FMCW radar sensor for driving assistance," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2018, pp. 1–6.

[17] W. Jiang, Y. Ren, Y. Liu, Z. Wang, and X. Wang, "Recognition of dynamic hand gesture based on mm-Wave FMCW radar micro-doppler signatures," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2021, pp. 4905–4909.

[18] R. Zhang and S. Cao, "Real-time human motion behavior detection via CNN using mmWave radar," *IEEE Sensors Lett.*, vol. 3, no. 2, pp. 1–4, Feb. 2019.

[19] Y. Wang, S. Wang, M. Zhou, Q. Jiang, and Z. Tian, "TS-I3D based hand gesture recognition method with radar sensor," *IEEE Access*, vol. 7, pp. 22902–22913, 2019.

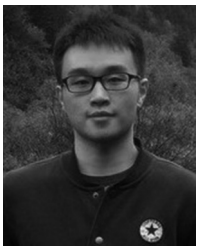
[20] M. Scherer, M. Magno, J. Erb, P. Mayer, M. Eggimann, and L. Benini, "TinyRadarNN: Combining spatial and temporal convolutional neural networks for embedded gesture recognition with short range radars," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10336–10346, Jul. 2021.

[21] P. Goswami, S. Rao, S. Bharadwaj, and A. Nguyen, "Real-time multi-gesture recognition using 77 GHz FMCW MIMO single chip radar," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, 2019, pp. 1–4.

[22] J.-T. Yu, L. Yen, and P.-H. Tseng, "mmWave radar-based hand gesture recognition using range-angle image," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, 2020, pp. 1–5.
[23] R. Huang, Z. Li, S. Wang, R. Wang, J. Li, and Z. Xu, "A RD-T network for hand gesture recognition based on millimeter-wave sensor," in *Proc. IEEE 5th Int. Conf. Signal Image Process. (ICSIP)*, 2020, pp. 308–312.
[24] Y. Wang, Z. Zhao, M. Zhou, and J. Wu, "Two dimensional parameters based hand gesture recognition algorithm for FMCW radar systems," in *Proc. Int. Conf. Wireless Satellite Syst.*, 2019, pp. 226–234.
[25] P.-H. Chen, Y.-J. Bai, and W. Jun, "Gesture recognition using LFMCW radar and convolutional neural network," *DEStech Trans. Comput. Sci. Eng.*, to be published.
[26] S. Hazra and A. Santra, "Short-range radar-based gesture recognition system using 3D CNN with triplet loss," *IEEE Access*, vol. 7, pp. 125623–125633, 2019.
[27] S. Hazra and A. Santra, "Radar gesture recognition system in presence of interference using self-attention neural network," in *Proc. 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2019, pp. 1409–1414.
[28] T. Sakamoto, X. Gao, E. Yavari, A. Rahman, O. Boric-Lubecke, and V. M. Lubecke, "Hand gesture recognition using a radar echo i–Q plot and a convolutional neural network," *IEEE Sensors Lett.*, vol. 2, no. 3, pp. 1–4, Sep. 2018.
[29] X. Yao, X. Shi, and F. Zhou, "Human activities classification based on complex-value convolutional neural network," *IEEE Sensors J.*, vol. 20, no. 13, pp. 7169–7180, Jul. 2020.
[30] W. Ye and H. Chen, "Human activity classification based on micro-doppler signatures by multiscale and multitask fourier convolutional neural network," *IEEE Sensors J.*, vol. 20, no. 10, pp. 5473–5479, May 2020.
[31] K. Rambach and B. Yang, "MIMO radar: Time division multiplexing vs. Code division multiplexing," in *Proc. Radar*, 2017, pp. 1–9.
[32] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Propag.*, vol. AP-34, no. 3, pp. 276–280, Mar. 1986.
[33] M. Schutz, C. Decroze, M. Lalande, and B. Lenoir, "Neural networks to increase range resolution of FMCW radar," *IEEE Sensors Lett.*, vol. 4, no. 8, pp. 1–4, Aug. 2020.
[34] M. Gall, M. Gardill, T. Horn, and J. Fuchs, "Spectrum-based single-snapshot super-resolution direction-of-arrival estimation using deep learning," in *Proc. IEEE German Microw. Conf. (GeMiC)*, 2020, pp. 184–187.
[35] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.
[36] H. Liu et al., "Real-time arm gesture recognition in smart home scenarios via millimeter wave sensing," *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 4, no. 4, pp. 1–28, 2020.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.
[38] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1120–1128.
[39] C. Trabelsi et al., "Deep complex networks," 2017, *arXiv:1705.09792*.
[40] N. Guberman, "On complex valued convolutional neural networks," 2016, *arXiv:1602.09046*.
[41] D. A. Brooks, O. Schwander, F. Barbaresco, J.-Y. Schneider, and M. Cord, "Complex-valued neural networks for fully-temporal micro-doppler classification," in *Proc. 20th Int. Radar Symp. (IRS)*, 2019, pp. 1–10.
[42] A. Vaswani et al., "Attention is all you need," 2017, *arXiv:1706.03762*.

Peijun Zhao received the B.E. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2017, and the Ph.D. degree from the Department of Computer Science, University of Oxford, Oxford, U.K., in 2022.

He is currently a Postdoctoral Associate with the Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA. His research interest lies in cyber–physical systems, especially, sensor data processing with machine-learning techniques for robotics and human-centered applications.



Chris Xiaoxuan Lu received the M.Eng. degree from Nanyang Technology University, Singapore, in 2015, and the Ph.D. degree from the University of Oxford, Oxford, U.K., in 2019.

He was a Postdoctoral Researcher with the University of Oxford. He is currently an Assistant Professor with the Department of Informatics, University of Edinburgh, Edinburgh, U.K. His research interest lies in cyber–physical systems, with the goal to drastically increasing their reliability, intelligence, and security in the wild.

Bing Wang received the Ph.D. degree from the Department of Computer Science, University of Oxford, Oxford, U.K., in 2022.

He is currently an Assistant Professor with the Department of Aeronautical and Aviation Engineering, Hong Kong Polytechnic University, Hong Kong. His research focuses on the cutting edge of robotics and autonomous systems research with the goal to enable human-level perception and world understanding on mobile robotics platforms in the wild.

Niki Trigoni received the D.Phil. degree from the University of Cambridge, Cambridge, U.K., in 2001.

She is a Professor with the Department of Computer Science, University of Oxford, Oxford, U.K. She became a Postdoctoral Researcher with Cornell University, Ithaca, NY, USA, from 2002 to 2004, and a Lecturer with Birkbeck College, London, U.K., from 2004 to 2007. In 2007, she joined the Department of Computer Science, University of Oxford, where she currently leads the Cyber Physical Systems Group. Her research interests lie in intelligent and autonomous sensor systems with applications in positioning, healthcare, environmental monitoring, and smart cities.

Prof. Trigoni is a Fellow of the Royal Academy of Engineering.

Andrew Markham received the B.Sc. and Ph.D. degrees from the University of Cape Town, Cape Town, South Africa, in 2004 and 2008, respectively.

He is a Professor with the Department of Computer Science, University of Oxford, Oxford, U.K. He works on sensing systems, with applications from wildlife tracking to indoor robotics to checking that bridges are safe. He works with the Cyber Physical Systems Group, University of Oxford. He designed novel sensors, investigated new algorithms (increasingly deep and reinforcement learning-based) and applied these innovations to solving new problems.